Near-optimal Balanced Reliable Broadcast and Asynchronous Verifiable Information Dispersal

Sourav Das, Ling Ren, Zhuolun Xiang

University of Illinois at Urbana-Champaign
{souravd2, renling, xiangzl}@illinois.edu

Abstract

In this paper, we present near-optimal asynchronous Byzantine reliable broadcast (RBC) protocols with balanced costs and an improved asynchronous verifiable information dispersal (AVID) protocol. Assuming the existence of collision-resistant hash functions, our RBC protocol broadcasts a message M among n nodes with total communication cost $O(n|M|+\kappa n^2)$ and per-node communication cost $O(|M|+\kappa n)$. In contrast, the state-of-the-art reliable broadcast protocol either has per-node cost $O(|M|+\kappa \log n)$, or has imbalanced costs where the broadcaster incurs $O(n|M|)$ while other nodes incur a communication cost of $O(|M|+\kappa n)$. We also present an error-free RBC protocol that makes no computational assumption and has total communication cost $O(n|M|+n^2 \log n)$ and per-node communication cost $O(|M|+n \log n)$. In contrast, the state-of-the-art error-free RBC protocol has total cost of $O(n|M|+n^3 \log n)$, and the broadcaster has imbalanced cost of $O(n|M|+n^2 \log n)$. We then use our new balanced RBC and additional techniques to design an asynchronous verifiable information dispersal (AVID) protocol with total dispersal cost $O(|M|+\kappa n^2)$, retrieval cost $O(|M|+\kappa n)$, and no trusted setup. In our AVID protocol, the client incurs a communication cost of $O(|M|+\kappa n)$ in comparison to $O(|M|+\kappa n \log n)$ of prior best. Moreover, each node in our AVID protocol incurs a storage cost of $O(|M|/n+\kappa)$ bits, in comparison to $O(|M|/n+\kappa \log n)$ bits of prior best. Finally, we present lower bound results on communication cost and show that our balanced RBC and AVID protocols have near-optimal communication costs – only an factor of $O(\kappa)$ or $O(\log n)$ gap from the lower bounds.

1 Introduction

Reliable broadcast (RBC) is a fundamental primitive in distributed computing [8], and has many applications such as fault-tolerant consensus and replication [25, 15, 18, 23], secure multiparty computation [22, 31], verifiable secret sharing [12], and distributed key generation [1, 21, 13]. The goal of RBC is to have a designated broadcaster send its input message and to have all nodes output the same message. Verifiable information dispersal (VID), introduced by Cachin and Tessaro [9], is another related primitive with emerging applications in fault-tolerant replication [15, 30]. VID lets a *client*, here on referred to as the *dispersing* client, disperse a message among a set of nodes during the *dispersal phase*, such that during the *retrieval phase* the message can be later retrieved by any node or any other client, which we refer to as the *retrieving* client. A VID protocol immediately implies a RBC protocol, where the broadcaster acts as the dispersing client and each node retrieves the data by acting as a retrieving client.

In this paper, we consider these two problems in asynchronous networks and we assume Byzantine faults that may deviate arbitrarily from the protocols. We consider the unauthenticated setting where the protocol does not use digital signatures. One of our RBC protocol is error-free and secure against any computationally unbounded adversary in all executions, while the other RBC protocol and the AVID assume collision resistant hash function.

Existing works on RBC. The first RBC protocol due to Bracha [8] has a total communication cost of $O(n^2|M|)$, where n is the number of protocol nodes and $|M|$ is the size of the broadcaster's message

Table 1: Comparison with existing RBC protocols. The following acronyms are used in the table; q-SDH: q-Strong Diffie-Hellman, DBDH: Decisional Bilinear Diffie-Hellman.

Scheme	Communication Cost (broadcaster)	Communication Cost (other node)	Communication Cost (total)	Rounds	Cryptographic Assumption	Setup
Bracha [8]	$O(n\|M\|)$	$O(n\|M\|)$	$O(n^2\|M\|)$	4	None (error-free)	None
Patra [27]	$O(n\|M\|+n^3 \log n)$	$O(\|M\|+n^3 \log n)$	$O(n\|M\|+n^4 \log n)$	11	None (error-free)	None
Nayak et al. [26]	$O(n\|M\|+n^2 \log n)$	$O(\|M\|+n^2 \log n)$	$O(n\|M\|+n^3 \log n)$	7	None (error-free)	None
This work	$O(\|M\|+n \log n)$	$O(\|M\|+n \log n)$	$O(n\|M\|+n^2 \log n)$	10	None (error-free)	None
Cachin-Tessaro [9]	$O(\|M\|+\kappa n \log n)$	$O(\|M\|+\kappa n \log n)$	$O(n\|M\|+\kappa n^2 \log n)$	4	Hash	None
Nayak et al. [26]	$O(\|M\|+\kappa n)$	$O(\|M\|+\kappa n)$	$O(n\|M\|+\kappa n^2)$	7	q-SDH+DBDH	Trusted
Das et al. [12]	$O(n\|M\|)$	$O(\|M\|+\kappa n)$	$O(n\|M\|+\kappa n^2)$	4	Hash	None
This work	$O(\|M\|+\kappa n)$	$O(\|M\|+\kappa n)$	$O(n\|M\|+\kappa n^2)$	5	Hash	None
Lower bound	$\Omega(\|M\|+n)$	$\Omega(\|M\|+n)$	$\Omega(n\|M\|+n^2)$	2 [2]	—	—

Table 2: Comparison with existing AVID protocols. The following accronyms are used in the table; DL: Discrete Logarithm, CRS: Common Reference String, q-SDH: q-Strong Diffie-Hellman.

Scheme	Dispersal Cost (client)	Dispersal Cost (total)	Retrieval Cost (total)	Storage Cost (total)	Cryptographic Assumption	Setup
Cachin-Tessaro [9]	$O(\|M\|+\kappa n \log n)$	$O(n\|M\|+\kappa n^2 \log n)$	$O(\|M\|+\kappa n \log n)$	$O(\|M\|+\kappa n \log n)$	Hash	None
Hendricks et al. [19]	$O(\|M\|+\kappa n^2)$	$O(\|M\|+\kappa n^3)$	$O(\|M\|+\kappa n^2)$	$O(\|M\|+\kappa n^2)$	Hash	None
Alhaddad et al. [3]	$O(\|M\|+\kappa n \log n)$	$O(\|M\|+\kappa n^2)$	$O(\|M\|+\kappa n \log n)$	$O(\|M\|+\kappa n \log n)$	DL	CRS
Alhaddad et al. [3]	$O(\|M\|+\kappa n)$	$O(\|M\|+\kappa n^2)$	$O(\|M\|+\kappa n)$	$O(\|M\|+\kappa n)$	q-SDH+Hash	Trusted
DisperseLedger [30]	$O(\|M\|+\kappa n \log n)$	$O(\|M\|+\kappa n^2)$	$O(\|M\|+\kappa n \log n)$	$O(\|M\|+\kappa n \log n)$	Hash	None
This work	$O(\|M\|+\kappa n)$	$O(\|M\|+\kappa n^2)$	$O(\|M\|+\kappa n)$	$O(\|M\|+\kappa n)$	Hash	None
Lower bound	$\Omega(\|M\|+n)$	$\Omega(\|M\|+n^2)$	$\Omega(\|M\|+n)$	$\Omega(\|M\|)$	—	—

in bits. Two decades later, it is improved by Cachin and Tessaro [9] to $O(n|M|+\kappa n^2 \log n)$, assuming a collision resistant hash function of output size $O(\kappa)$. In both of these RBC protocols, every node, including the broadcaster, incurs the same asymptotic communication cost. Here on, we say such a RBC protocol has a *balanced* communication cost. The state-of-the-art asynchronous RBC protocol in terms of total communication cost is due to Das et al. [12], which has a total communication cost of $O(n|M|+\kappa n^2)$ and requires no trusted setup. However, it has an *unbalanced* communication cost. The cost of the broadcaster is approximately n times higher than that of other nodes, leading to a bottleneck at the broadcaster. The state-of-the-art error-free RBC protocol [26] has a total communication cost of $O(n|M|+n^3 \log n)$ and is also unbalanced. We provide a detailed comparison in Table 1 and discuss other related work in detail in §6.

Existing works on Asynchronous VID (AVID). Cachin and Tessaro [9] presented the first AVID protocol, with a total communication cost of $O(n|M|+\kappa n^2 \log n)$ during the dispersal phase and $O(|M|+\kappa n \log n)$ during the retrieval phase. It is then improved by Hendricks et al. [19], and very recently, by Alhaddad et al and Yang et al. [30, 3] to $O(|M|+\kappa n^2)$ for dispersal and $O(|M|+\kappa n \log n)$ for retrieval phase. In their protocols, both the dispersing client and the retrieving client incur a cost of $O(|M|+\kappa n \log n)$, and each node incurs $O(|M|/n + \kappa \log n)$ storage cost. We summarize the existing works on AVID in Table 2 and describe them in more detail in §6.

Our contributions. Our first contribution is two balanced RBC protocols. The first protocol assumes collision resistant hash function, has the same total communication cost of $O(n|M|+\kappa n^2)$ as the state-of-the-art [12], but additionally achieves balanced communication cost of $O(|M|+\kappa n)$ at every node, including the broadcaster. Our main technique to achieve balanced communication cost is to use an additional round to interaction between nodes to help them reconstruct the potential input of the broadcaster without having the broadcaster to directly send the input to all nodes. Our second RBC protocol is error-free, balanced, and has total communication cost to $O(n|M|+n^2 \log n)$, compared to $O(n|M|+n^3 \log n)$ communication of

current art [26]. Our error-free RBC protocol builds on top of the recent synchronous error-free Byzantine agreement (BA) protocol of Chen [11]. We make several subtle and important modifications to their BA protocol to accommodate asynchrony and achieve balanced cost (see §A).

Our second contribution is an AVID protocol that does not require any trusted setup and has a communication cost of $O(|M|+\kappa n^2)$ during the dispersal phase. Moreover, in our AVID protocol, both dispersing and retrieving clients incur a communication cost of $O(|M|+\kappa n)$. Similar to existing AVID protocols [9, 19, 30, 3], during the dispersal phase, the dispersing client uses error correction code to encode the message into n symbols and send a symbol to each node. Unlike existing protocols, we do not use a Merkle tree for verification; instead, the dispersing client reliably broadcasts hashes of all symbols that the nodes use for verification. As a result, the dispersing client in our AVID protocol incurs a communication cost of $O(|M|+\kappa n)$. We also reduce the per node storage to $O(|M|/n + \kappa)$ from $O(|M|/n + \kappa \log n)$, and the communication cost of the retrieval phase to $O(|M|+\kappa n)$ from $O(|M|+\kappa n \log n)$, by designing a novel retrieval phase.

Our third contribution is two lower bound results. We prove that for any deterministic RBC protocol, each node incurs a communication cost of $\Omega(|M|+n)$. We also prove that in any deterministic AVID protocol, the dispersal phase has a communication cost of $\Omega(|M|+n^2)$ and the retrieval phase has a communication cost of $\Omega(|M|+n)$. Hence, all of our protocols above have near-optimal communication costs – only an factor of $O(\log n)$ or $O(\kappa)$ gap from the lower bounds.

Paper organizations. The rest of the paper is organized as follows. In §2 we provide the necessary background. In §3 we discuss our balanced RBC protocols where the broadcaster has the same bandwidth cost as other nodes. In §4 we describe our improved AVID protocol. In §5 we show lower bounds on the communication cost of RBC and AVID. We discuss related work in §6 and conclude in §7. We provide detailed description of our error-free RBC and its analysis in Appendix A.

2 System Model and Preliminaries

2.1 System Model

We consider a network of n nodes where every pair of nodes is connected via a pairwise authenticated channel. We consider the presence of a malicious adversary $\mathcal{A}$ that can corrupt up to t nodes in the network. The corrupted nodes can behave arbitrarily, and we call a node honest if it remains non-faulty for the entire protocol execution. We assume the network is asynchronous, i.e., $\mathcal{A}$ can arbitrarily delay any message but must eventually deliver all messages sent between honest nodes. A protocol is error-free if it is secure against any computationally unbounded adversary in all executions.

We use $|S|$ to denote the size of a set S. Let $\mathbb{F}$ be a finite field. For any integer a, we use $[a]$ to denote the set $\{1, 2, \ldots, a\}$. We use κ to denote the size of the output of the collision-resistant hash function. Naturally, we assume that $\kappa > \log n$.

2.2 Problem Formulations

Definition 1 (Reliable Broadcast [8])**.** A protocol for a set of nodes $\{1, \ldots, n\}$, where a designated broadcaster holds an input M, is a reliable broadcast (RBC) protocol, if the following properties hold

- *Agreement:* If an honest node outputs a message M' and another honest node outputs M'', then $M' = M''$.
- *Validity:* If the broadcaster is honest, all honest nodes eventually output the message M.
- *Totality:* If an honest node outputs a message, then every honest node eventually outputs a message.

A VID protocol has two functions: `DISPERSE`(M), which a client invokes to disperse a message M to n nodes, and `RETRIEVE`, which a (possibly different) client invokes to retrieve the message M. Clients invoke `DISPERSE` and `RETRIEVE` for a particular instance of VID, identified by an instance tag. For simplicity, in the paper we will focus on a single instance of an VID and omit the instance tag.

Definition 2 (Verifiable Information Dispersal [9])**.** A verifiable information dispersal (VID) scheme for a message M consists of a pair of protocols `DISPERSE` and `RETRIEVE` which satisfy the following requirements:

- *Termination:* If an honest client invokes `DISPERSE`(M) and no other client invokes `DISPERSE` on the same instance, then every honest node eventually finishes the dispersal phase.
- *Agreement*: If any honest node finishes the dispersal phase, all honest nodes eventually finish the dispersal phase.
- *Availability*: If an honest node has finished the dispersal phase, and some honest client initiates `RETRIEVE`, then the client eventually reconstructs some message M'.
- *Correctness*: If an honest node has finished the dispersal phase, then honest clients always reconstruct the same message M' when invoking `RETRIEVE`. Furthermore, if an honest client invoked `DISPERSE`(M) and no other client invokes `DISPERSE` on the same instance, then $M' = M$.

In this paper, we will propose protocols solving these two problems under asynchronous networks, and also prove corresponding lower bounds for deterministic protocols.

Definition 3 (Communication Cost)**.** The (total) communication cost of a protocol measures the total number of bits sent by all honest protocol nodes during the execution of the protocol.

In addition to the standard communication cost above which measures the total cost of a protocol, we also measure the cost for each honest protocol node, as the per-node communication cost defined below.

Definition 4 (Per-node Communication Cost)**.** The communication cost of any honest protocol node p running a protocol measures the number of bits sent by p, and the number of bits p received from any other honest node, during the execution of the protocol. We say the protocol has per-node communication cost of C, if every honest node has communication cost at most C asymptotically.

For instance, in the RBC protocol of Das, Xiang and Ren [12], the broadcaster incurs cost $O(n|M|)$ and any other node incurs cost $O(|M|+\kappa n)$, therefore the per-node communication cost of the protocol is $O(n|M|)$. Note that the total communication cost of a protocol equals the sum of communication costs of all honest nodes asymptotically.

Definition 5 (Balanced Protocol)**.** We say a protocol is balanced, if the per-node communication cost is $O(C/n)$ where C is the total communication cost of the protocol; otherwise, the protocol is unbalanced.

For instance, in the RBC protocol of Das, Xiang and Ren [12] is *unbalanced*, since the per-node communication cost of the protocol is $O(n|M|)$ and the total communication cost of the protocol is $O(n|M|+\kappa n^2)$. In contrast, our RBC protocol in §3.1 is *balanced*, as it has per-node communication cost $O(|M|+\kappa n)$ and total communication cost $O(n|M|+\kappa n^2)$.

2.3 Primitives

Error Correcting Code. We use error correcting codes. For concreteness, we will use the standard Reed-Solomon (RS) codes [28]. A (m, k) RS code in Galois Field $\mathbb{F} = \mathrm{GF}(2^a)$ with $m \leq 2^a - 1$, encodes k data symbols from $\mathrm{GF}(2^a)$ into a codeword of m symbols from $\mathrm{GF}(2^a)$. Let $\mathsf{RSEnc}(M, m, k)$ be the encoding algorithm. Briefly, the RSEnc takes as input a message M consisting of k symbols, treats it as a polynomial of degree $k-1$ and outputs m evaluations of the corresponding polynomial.

Let $\mathsf{RSDec}(k, r, T)$ be the RS decoding procedure. RSDec takes as input a set of symbols T (some of which may be incorrect), and outputs a degree $k-1$ polynomial, i.e., k symbols, by correcting up to r errors (incorrect symbols) in T. It is well-known that RSDec can correct up to r errors in T and output the original message provided that $|T| \geq k + 2r$ [24]. Concrete instantiations of RS codes include the Berlekamp-Welch algorithm [29] and the Gao algorithm [17].

Online Error Correction. Both of our RBC protocols and AVID protocol use the Online-Error-Correction (OEC) protocol introduced by Ben-Or [4]. The OEC takes a set T consisting of tuples (j, a_j) where j is an index $j \in [n]$ and a_j is a symbol of a Reed-Solomon codeword. The OEC algorithm then tries to decode a message M such that Reed-Solomon encoding of M matches with at least $2t+1$ elements in T. More specifically, the OEC algorithm performs up to t trials of reconstruction, and during the r-th trial, it uses $2t+r+1$ elements in T to decode. If the reconstructed message M' has the matches with at least $2t+1$ tuples in T, the OEC algorithm successfully outputs the message; otherwise, it waits for one more symbol and tries again. We summarize the OEC algorithm in Algorithm 1. The OEC algorithm is error-free and information-theoretically secure against any adversary that corrupts up to t symbols among a total of $n \geq 3t+1$ symbols.

Algorithm 1 Information Theoretic Online Error-correcting (IT-OEC) protocol

1: **Input:** T
2: **for** $0 \leq r \leq t$ **do** *// online Error Correction*
3: Wait till $|T| \geq 2t+r+1$
4: Let $p_r(\cdot) := \mathsf{RSDec}(t+1, r, T)$
5: **if** $2t+1$ elements $(j, a) \in T$ satisfy $p_r(j) = a$ **then**
6: let M be the coefficients of $p_r(\cdot)$
7: return M

Collision-resistant Hash Function. We use a cryptographic collision-resistant hash function hash, which guarantees that a computationally bounded adversary cannot come up with two inputs that hash to the same value, except for a negligible probability.

3 Near-optimal Balanced Reliable Broadcast

In this section, we present our balanced reliable broadcast protocols. We first present a hash-based balanced reliable broadcast protocol named BalRBC in Algorithm 2 and describe it in §3.1. We then briefly introduce our balanced error-free RBC protocol named BalEFRBC in §3.2, and leave its details to Appendix A. We only discuss the hash-based RBC protocol in the main paper due to its simplicity and space restrictions. Both protocols are balanced as per definition 4. Our BalEFRBC protocol achieves better total and per-node communication cost than state-of-the-art [26], while our BalRBC protocol matches the state-of-the-art *unbalanced* RBC protocol in terms of total cost [12].

3.1 Balanced Hash-based Reliable Broadcast

Challenges and our approaches. The state-of-the-art RBC protocol of Das, Xiang, and Ren [12] crucially uses the fact that the broadcaster sends its input message to all nodes at the start of the protocol. In their protocol, roughly speaking, nodes run Bracha's RBC on the hash digest of their message received from the broadcaster, and only exchange coded symbols of the message to reduce the communication cost to $O(n|M|+\kappa n^2)$. To ensure correctness, a node should RBC the digest and exchange the coded symbols of the same message. It is straightforward in the protocol of Das, Xiang, and Ren [12] since nodes directly receive the message from the broadcaster, which however incurs a cost of $O(n|M|)$ at the broadcaster. To reduce the cost, a natural idea would be let the broadcaster only send coded symbols of its message. Then, some kinds of proof would be necessary to convince the nodes that the coded symbols are consistent with each other, otherwise the nodes can no longer run the erasure decoding protocol to recover the message. In fact, Cachin and Tessaro [9] obviates the need for the broadcaster to send its input to all via this approach. Specifically, the broadcaster encodes its input using an Error Correcting Code, computes a Merkle tree on the encoded symbol, and to each node, sends one encoded symbol and the corresponding Merkle path. A consequence of using a Merkle tree is that the resulting protocol has a communication cost of $O(n|M|+\kappa n^2 \log n)$, since the

Algorithm 2 BalRBC protocol for long messages

1: *// only broadcaster node*
2: **input** M
3: Let $h := \mathsf{hash}(M)$
4: Let $[m_1, m_2, \ldots, m_n] := \mathsf{RSEnc}(M, n, t+1)$
5: **send** $\langle \texttt{PROPOSE}, m_j \rangle$ to node j for each $j \in [n]$

// each node i
6: Let $M := \perp$
7: **upon** receiving the first $\langle \texttt{PROPOSE}, m_i \rangle$ from the broadcaster **do**
8: **send** $\langle \texttt{SHARE}, m_i \rangle$ to all nodes

9: For the first $\langle \texttt{SHARE}, m_j^* \rangle$ received from node j, add (j, m_j^*) to T *// T initialized as* $\{\}$
10: Run IT-OEC on the set T
11: Let M' be the output of IT-OEC(T)

12: **if** $P(M') = true$ **then** *// $P(\cdot)$ is an external predicate that returns true or false. See protocol description for more details.*
13: Let $h := \mathsf{hash}(M')$
14: **send** $\langle \texttt{ECHO}, m_j, h \rangle$ to node j for each $j \in [n]$ where m_j is the j-th symbol of $\mathsf{RSEnc}(M', n, t+1)$

15: **upon** receiving $2t+1$ $\langle \texttt{ECHO}, m_i, h \rangle$ for the same m_i, h and not having sent a `READY` message **do**
16: **send** $\langle \texttt{READY}, m_i, h \rangle$ to all

17: **upon** receiving $t+1$ $\langle \texttt{READY}, *, h \rangle$ for the same h and not having sent a `READY` message **do**
18: Wait for $t+1$ matching $\langle \texttt{ECHO}, m_i', h \rangle$
19: **send** $\langle \texttt{READY}, m_i', h \rangle$ to all

20: For the first $\langle \texttt{READY}, m_j^*, h \rangle$ received from node j, add (j, m_j^*) to T_h *// T_h initialized as* $\{\}$
21: Run IT-OEC on the set T_h
22: Let M'' be the output of IT-OEC(T_h)
23: **output** M'' and **return**

size of the Merkle path is $O(\kappa \log n)$ and there are all-to-all message exchanges with Merkle path attached. Similarly, Alhaddad et al. [3] uses a trusted setup-based constant size polynomial commitment scheme where the proof has size $O(\kappa)$ instead of the Merkle tree, to design a balanced RBC with a total communication cost of $O(n|M|+\kappa n^2)$. Omitting the Merkle tree or the polynomial commitment in a naive manner introduces the challenge that we can no longer run the erasure decoding protocol used by [9, 3] as there does not exist a way to distinguish between an incorrect symbol from a correct one.

Our observation is that, there is a simple way to balance the cost without attaching proofs with the coded symbols. The idea is to add one more communication round for the nodes to exchange their symbols received from the broadcaster, and try Information Theoretic Online Error Correction (IT-OEC) to reconstruct the broadcaster's message. If the broadcaster is honest, then IT-OEC can always recover the broadcaster's message. In case of a malicious broadcaster, an honest node may not recover the message from IT-OEC in the beginning; but rest of our protocol guarantees that if any honest node output for the RBC, then there are enough honest nodes holding correct symbols that will send the symbols to all nodes for reconstruction. In fact, this simple idea turns out to be very useful, as we can also apply it to our error-free RBC in §3.2 for balancing the cost.

Protocol description. In order to reduce the cost of the broadcaster node, our protocol BalRBC first lets the broadcaster encode its message M into n symbols using a $(n, t+1)$ Reed-Solomon code (line 4) and only send the i-th symbol to node i together with the hash digest of the message M. In particular, let $[m_1, m_2, \ldots, m_n] = \mathsf{RSEnc}(M, n, t)$ be the RS encoding of M. Then, to node i, the broadcaster sends the

message $\langle \texttt{PROPOSE}, m_i \rangle$ (line 5). Note that due to properties RS code, each symbol has size $|M|/(t+1)$, and therefore the cost of the broadcaster is reduced to $O(n \cdot (|M|/(t+1) + \kappa)) = O(|M| + \kappa n)$ where κ is the size of the hash digest.

Next, each node i upon receiving the $\langle \texttt{PROPOSE}, m_i \rangle$ message from the broadcaster sends the $\langle \texttt{SHARE}, m_i \rangle$ to all nodes (line 7-8). When a node receives a SHARE message from other nodes, it adds the corresponding symbol to the set T. Once enough symbols are collected, nodes use the Online Error Correcting (OEC) algorithm (line 10-11) to decode the message. As described in 2, intuitively, the OEC algorithm performs up to t trials of reconstruction, and during the r-th trial, a node uses $2t + r + 1$ symbols to decode. If the reconstructed message M' has the matches with at least $2t + 1$ tuples in T, a node successfully reconstructs the message; otherwise, it waits for one more symbol and tries again.

Once a node successfully reconstructs the message M', the rest of the protocol is similar to the four-round RBC of Das, Xiang and Ren [12, Algorithm 4]. Similar to Das et al., we also add an *external predicate* $P(\cdot)$ (line 12) to strengthen the validity guarantee of our BalRBC, so that any honest node only output M such that $P(M) = true$. This external validity check is useful for many application of RBC, including verifiable secret sharing [12] and AVID. In fact, our AVID protocol in §4 will use such RBC with external validity check. For standard RBC, $P(\cdot)$ always returns true.

Briefly, after checking the predicate $P(\cdot)$, nodes send ECHO messages with their symbols and the hash digest to all nodes (line 14). Also, nodes send READY messages once $2t + 1$ matching ECHO messages are collected (line 15-16) or upon receiving $t + 1$ READY messages (line 17). Note that each node needs to wait for $t + 1$ matching ECHO messages to learn the symbol to be attached in the READY message (line 18-19). Finally, nodes use the OEC algorithm to reconstruct the broadcaster's message once receiving enough READY messages of the same hash digest.

We next analyze the properties of our BalRBC protocol and its performance.

Lemma 1. *Assuming a collision resistant hash function, if an honest node sends* $\langle \texttt{READY}, m_i, h \rangle$ *where* $h = \mathsf{hash}(M)$*, then* m_i *is the* i^{th} *symbol of* $\mathsf{RSEnc}(M, n, t+1)$*, and furthermore, no honest node sends a* READY *message for a different hash* $h' \neq h$*.*

Proof. First, no two honest nodes send READY messages for different hash digests, due to quorum intersection of the ECHO messages same as the Bracha's RBC. Now we show if an honest node sends $\langle \texttt{READY}, m_i, h \rangle$ where $h = \mathsf{hash}(M)$, then m_i is the i^{th} symbol of $\mathsf{RSEnc}(M, n, t+1)$. Note that an honest node i sends $\langle \texttt{READY}, m_i, h \rangle$ for $h = \mathsf{hash}(M)$ only upon receiving at least $t+1$ matching $\langle \texttt{ECHO}, m_i, h \rangle$. At least one of these ECHO message is from an honest node h. Before the honest node h sends the ECHO message, it successfully reconstructed the message M' whose hash digest equals h. Then, by the collision resistance property of the underlying hash function, $M' = M$ and m_i is the i^{th} symbol of $\mathsf{RSEnc}(M, n, t+1)$. □

Lemma 2. *If an honest node* i *receives* $t + 1$ READY *messages with a matching hash* h*, then node* i *will eventually receive* $t + 1$ *matching* $\langle \texttt{ECHO}, m_i, h \rangle$ *messages and hence send* $\langle \texttt{READY}, m_i, h \rangle$*.*

Proof. Let j be the first honest node that sends $\langle \texttt{READY}, *, h \rangle$ message to all. Then, node j must have received at least $2t + 1$ ECHO messages with matching h, among which at least $t + 1$ are from honest nodes. Hence, node i will eventually receive $t + 1$ $\langle \texttt{ECHO}, m_i, h \rangle$ messages from these honest nodes. □

Theorem 1 (Totality and Agreement)**.** *If an honest node outputs a message, then every honest node eventually outputs a message. If an honest node outputs a message* M' *and another honest node outputs* M''*, then* $M' = M''$*.*

Proof. An honest node outputs a message M only upon receiving at least $2t + 1$ READY messages with a matching hash $h = \mathsf{hash}(M)$. At least $t + 1$ of them are sent by an honest node. Hence, all honest nodes will receive at least $t + 1$ READY messages with hash h. By lemma 2, eventually all honest nodes will send READY messages with hash h. Hence, all honest nodes will receive READY messages from all other honest nodes. Furthermore, due to Lemma 1, all these READY message contain correct symbols from the codeword $\mathsf{RSEnc}(M, n, t + 1)$. Thus, every honest node will eventually output M such that $h = \mathsf{hash}(M)$. □

Theorem 2 (Validity)**.** *If the broadcaster node is honest, has an input M, and $P(M) = true$, then all honest nodes eventually output the message M.*

Proof. When the broadcaster is honest and has input M, it sends the correct symbols and hash to all nodes. Then, all honest nodes send the `SHARE` messages with the correct symbols. Thus, after receiving all `SHARE` message from honest nodes, any honest node can reconstruct M due to OEC and collision resistance of the hash. Also, the predicate $P(M) = true$ at all honest nodes, so at least $2t + 1$ honest nodes will send `ECHO` messages with identical $h = \mathsf{hash}(M)$. Hence, all honest nodes will eventually send `READY` messages for h. By lemma 1 no honest node will send `READY` message for $h' \neq h$. As a result, all honest node will receive at least $2t + 1$ `READY` message for h with valid symbols in it, which is sufficient to recover M. □

Next, we will analyze the communication complexity of the protocol.

Theorem 3 (Performance)**.** *Assuming existence of a collision resistant hash function whose outputs are κ bits long, Algorithm 2 solves RBC with total communication cost of $O(n|M|+\kappa n^2)$, and per-node communication cost of $O(|M|+\kappa n)$.*

Proof. In algorithm 2 the broadcaster sends a single `PROPOSE` to all other nodes. Moreover, each honest node sends a single `SHARE`, `ECHO` and `READY` message. Each message in Algorithm 2 is $O(|M|/n+\kappa)$ bits long, since $|m_i|= |M|/(t+1)$ and hash outputs are κ bits long. Hence, each node incurs a per-node communication cost of $O(|M|+n\kappa)$. Hence, the total communication cost is $O(n|M|+\kappa n^2)$. □

3.2 Balanced Error-Free Reliable Broadcast

In this section, we will only briefly introduce our BalEFRBC protocol for brevity. Our protocol is heavily inspired by the recent error-free synchronous Byzantine agreement protocol named COOL [11]. We extend the COOL protocol [11] to obtain an error-free asynchronous RBC protocol with per-node communication cost $O(|M|+n \log n)$. The full protocol and correctness proofs are presented in Appendix A. We make the following three major changes to obtain our BalEFRBC protocol.

1. *Triggering the next message upon receiving sufficient messages asynchronously, instead of receiving all messages from the previous synchronous round.* The COOL protocol is a synchronous protocol that proceeds in lock-step rounds, so it contains several steps where nodes wait for all messages from the previous round before taking their next step. For instance, a node will send an indicator message for 1 if it receives $n - t$ 1-indicators in the previous round; otherwise it sends an indicator for 0. However, under asynchrony, the node cannot expect to receive all messages, since a slow honest node is indistinguishable from a Byzantine node. Therefore, we need to change the triggering event to receiving enough messages asynchronously. For instance, the above example is changed to the following: a node sends 1-indicator upon receiving $n-t$ 1-indicators and sends 0-indicator upon receiving $t+1$ 0-indicators. Since there are n nodes in total and each node can send one indicator, the above two conditions will not hold simultaneously. Moreover, if the original synchronous protocol relies on the fact that a node receives enough indicators, then the new asynchronous protocol preserves the same property since the node only triggers the message event after receiving enough indicators.

2. *Replacing the 1-bit asynchronous BA with 1-bit Bracha's RBC.* The COOL protocol uses a synchronous binary BA protocol for all the nodes to agree on whether there are enough honest nodes holding the correct coded message symbols in order to recover the message. Our BalEFRBC also requires a similar step under asynchrony. However, we cannot use an asynchronous binary BA to construct an error-free RBC, because any asynchronous BA has to be randomized due to the FLP impossibility result [16]. Instead, we use the 1-bit Bracha's RBC [8], which is error-free, as follows. When a node inputs 1 (or 0) to the synchronous BA in the COOL protocol, we let the node send an `ECHO` message for 1 (or 0) in the 1-bit Bracha's RBC. As a result, the 1-bit Bracha's RBC guarantees agreement among the nodes on whether they should reconstruct the message or simply output a default message $\perp$. Moreover, as we will explain in Appendix A, if one honest node outputs 1 in the 1-bit Bracha's RBC, then every honest node will be able to reconstruct and output the same message.

3. *Balancing the broadcaster's cost by the technique of §3.1.* The RBC protocol obtained after the above two changes is still not balanced, since the straightforward transformation from agreement to broadcast asks the broadcaster to send the entire message in the first step of the protocol, leading to a cost of at least $\Omega(n|M|)$ at the broadcaster. Therefore, we apply the technique of §3.1, which ensures that broadcaster incurs a communication cost of $O(|M|+n\log n)$.

4 Improved Asynchronous Verifiable Information Dispersal

We next describe our improved AVID protocol and summarize it in Algorithm 3. As mentioned in §2, the protocol consists of two phases: *dispersal* phase, invoked by the dispersing client who wants to reliably store its message M among n protocol nodes; and *retrieval* phase, invoked by any retrieving client (need not be the one who initiated the dispersal phase), who wants to retrieve the message M.

4.1 Challenges and Our Approaches.

In the state-of-the-art AVID protocol that does not rely on a trusted setup, both the dispersing and retrieving clients incur a communication cost of $O(|M|+\kappa n\log n)$. This is because the dispersing client sends a Merkle path to each node and an encoded symbol of its input message. Nodes use the Merkle path to check the consistency of the encoded symbol they receive from the dispersing client and also prove the consistency of the symbol to a retrieving client during the retrieval phase. As a result, each node needs to store the Merkle path throughout, thus incurs a storage cost of $O(|M|/n+\kappa\log n)$. Alhaddad et al. [3] improves the communication and storage cost to $O(|M|+\kappa n)$ and $O(|M|/n+\kappa n)$, respectively, using trusted setup phase polynomial commitments. Omitting the Merkle tree or the polynomial commitment in a naive manner introduces the challenge that, during the dispersal phase, nodes can not check the consistency of the symbol received from the dispersing client. Moreover, the non-existence of a consistency proof allows Byzantine nodes to equivocate to different retrieving clients. As a result, different retrieving clients may output different messages, hence violating the Correctness property.

Briefly, we address these challenges with the following ideas. First, we replace the Merkle tree with a vector of hashes of the encoded symbols and then let the dispersing client reliably broadcast the entire vector of hashes to every node. The nodes then use this vector to check the consistency of the symbols they receive from the dispersing client. Since the size of the vector is $O(\kappa n)$, using the BalRBC protocol with balanced cost from §3, the total communication of all nodes and the dispersing client during the RBC protocol is $O(\kappa n^2)$ and $O(\kappa n)$, respectively. Note that, during the retrieval phase, the retrieving client does not have the vector of hashes thus can not directly validate the symbols it receives from other nodes. Furthermore, asking each sender to send the entire vector would result in $O(\kappa n^2)$ communication cost. We address this by first letting the client retrieve the vector of hashes using the Online Error Correcting (OEC) protocol. The client then uses the retrieved vector for the rest of the retrieval phase. As we discuss next, our approach, in addition to reducing the communication cost of the retrieval phase to $O(|M|+\kappa n)$, also reduces the per node storage cost to $O(|M|/n+\kappa)$ bits.

4.2 Design of AVID

In the dispersal phase, first, the dispersing client encodes the message M using a $(n, t+1)$ Reed-Solomon code (line 2). Let $M' = [m_1, m_2, \ldots, m_n]$ be the encoded message with n symbols. The dispersing client then computes a vector $H = [h_1, h_2, \ldots, h_n]$ where the i-th element $h_i = \mathsf{hash}(m_i)$ (line 3). The dispersing client then sends the i-th symbol m_i to the i-th node. Additionally, the dispersing client reliably broadcasts H using our BalRBC protocol from §3 (line 4). During the BalRBC, as per the predicate, the i-th node checks whether the i-th element of the hash vector that is being reliably broadcast, is equal to the hash of the symbol it received from the dispersing client (line 5-7). Let H be the output of the validated RBC. Each node then encodes H using a $(n, t+1)$ Reed-Solomon code; let $H' = [h'_1, h'_2, \ldots, h'_n]$ be the output of the

Algorithm 3 Pseudocode for AVID

// the dispersing client invokes `DISPERSE`*(M)*
1: **input** M
2: Let $M' := [m_1, m_2, \ldots, m_n] := \mathsf{RSEnc}(M, n, t+1)$
3: Let $H := [h_1, h_2, \ldots, h_n] := [\mathsf{hash}(m_1), \mathsf{hash}(m_2), \ldots, \mathsf{hash}(m_n)]$
4: Send m_i to node i for each $i \in [n]$, and invoke BalRBC(H) with predicate $P(\cdot)$ described below

// additional predicate $P(\cdot)$ for node i to check in line 12 of BalRBC
5: **procedure** $P(H)$
6: **upon** receiving m_i from the dispersing client **do**
7: **return** true iff $\mathsf{hash}(m_i) = H[i]$

// code for node i during the dispersal phase
8: Wait till BalRBC($\cdot$) terminate
9: Let $H = [h_1, h_2, \ldots, h_n]$ be the output of BalRBC($\cdot$)
10: Let $h = \mathsf{hash}(H)$
11: $[h'_1, h'_2, \ldots, h'_n] := \mathsf{RSEnc}(H, n, t+1)$
12: Output and store $\langle m_i, h'_i, h\rangle$ for the dispersal phase *// Use $m_i = \perp$ if it does not receive m_i such that $h_i = \mathsf{hash}(m_i)$ from the client.*

// the retrieving client invokes `RETRIEVE`
13: **send** $\langle$`RETRIEVE`$\rangle$ to all nodes

// retrieving H
14: Let $T_h := \{\}$ and $T_M := \{\}$
15: For every $\langle$`HASH`$, h'_j, h\rangle$ received from node j, add (j, h'_j) to T_h
16: For every $\langle$`SYMBOL`$, m_j\rangle$ received from node j, add (j, m_j) to T_M
17: Run IT-OEC using T_h
18: Let $H :=$ IT-OEC(T_h)

// retrieving M after retrieving H
19: **for** each $(j, a) \in T_M$ **do**
20: **if** $\mathsf{hash}(a) = H[j]$ **then**
21: add (j, a) to T
22: Wait till $|T| = t+1$
23: Interpolate T as a degree-t polynomial
24: Let M' be the interpolated polynomial evaluated at every element in $[n]$
25: **if** $\exists j \in [n]$ such that $\mathsf{hash}(M'[j]) \neq H[j]$ **then**
26: **output** $\perp$ and **return**
27: **else**
28: **output** $\mathsf{RSDec}(t+1, 0, M')$ and **return**

// code for node i during the retrieval phase
29: **upon** receiving $\langle$`RETRIEVE`$\rangle$ from the retrieving client **do**
30: Wait till the dispersal phase outputs $\langle m_i, h'_i, h\rangle$
31: send $\langle$`HASH`$, h'_i, h\rangle$ to the retrieving client
32: **if** $m_i \neq \perp$ **then**
33: send $\langle$`SYMBOL`$, m_i\rangle$ to the retrieving client

Reed-Solomon encoding (line 11). Also, let $h = \mathsf{hash}(H)$. At the end of the dispersal phase, the i-th node outputs $\langle m_i, h'_i, h\rangle$ where $m_i = \perp$ if the i-th node did not receive a valid symbol from the dispersing client.

The main idea of the retrieval phase is to let the retrieving client first recover the vector H and then use it to validate symbols sent by nodes. More specifically, during the retrieval phase, the retrieving client sends

RETRIEVE request to all nodes (line 13). Upon receiving RETRIEVE request from the retrieving client, each node waits till the dispersal phase terminates (line 30). The i-th node then sends the message $\langle \texttt{HASH}, h'_i, h \rangle$ to the retrieving client (line 31). Additionally, if node i received a symbol m_i during the dispersal phase such that $\mathsf{hash}(m_i) = H[i]$, it sends a $\langle \texttt{SYMBOL}, m_i \rangle$ to the retrieving client (line 32-33).

The retrieving client upon receiving a message $\langle \texttt{HASH}, h_j, h \rangle$ from node j adds them to a set T_h (line 15). Additionally, for every $\langle \texttt{SYMBOL}, m_j \rangle$ message it receives, it adds it to the set T_M. The retrieving client then uses T_h and the standard online error correction to recover H (line 17-18). After recovering H, the retrieving client uses it to retrieve the message. In particular, for every tuple $(j, a) \in T_M$, it first checks whether $\mathsf{hash}(a) = H[j]$ and adds the tuple (j, a) to the set T (line 19-21).

The retrieving client waits till $|T| = t+1$ and then interpolates the tuples in T into a polynomial of degree at most t (line 22-23). Let M' be the interpolated polynomial. The client then checks if there exists any $j \in [n]$ such that $\mathsf{hash}(M[j]) \neq H[j]$. If such j exists, then the client outputs $\perp$ and returns. Otherwise, the client outputs the Reed-Solomon decoding of M' (line 28).

4.3 Analysis of AVID

We next analyze the properties of our AVID protocol and its performance.

Theorem 4 (Termination and Agreement)**.** *If an honest dispersing client invokes* DISPERSE(M) *and no other client invokes* DISPERSE *on the same instance, then every honest node eventually finishes the dispersal phase. If any honest node finishes the dispersal phase, all honest nodes eventually finish the dispersal phase.*

Proof. An honest dispersing client sends the correct symbols $[m_1, m_2, \ldots, m_n] = \mathsf{RSEnc}(M, n, t+1)$, and reliably broadcasts the hash vector $H = [\mathsf{hash}(m_1), \mathsf{hash}(m_2), \ldots, \mathsf{hash}(m_n)]$. By the Validity property of the RBC, the RBC will terminate at all honest nodes. Hence, every honest node will finish the dispersal phase.

A nodes terminates the dispersal phase if and only if the RBC protocol terminates. Thus, by the Totality property of the RBC every node will terminate the RBC and thus terminate the dispersal phase. □

Lemma 3. *If the dispersal phase terminates at an honest node, then every honest node will output the same vector of hashes $H = [h_1, h_2, \ldots, h_n]$ for RBC. Furthermore, at least $t+1$ honest nodes have received a symbol that matches with the corresponding location of H.*

Proof. Nodes terminate the dispersal phase if and only if the RBC protocol terminates. Thus by the Totality and Agreement property of the RBC every node will receive the same message H. Furthermore, when any honest node finishes the RBC, it has received READY messages from at least $2t+1$ nodes, among which at least $t+1$ are honest. Thus, at least one honest node receives ECHO messages from at least $2t+1$ nodes, among which at least $t+1$ are honest. Before these honest nodes send ECHO messages, they have the predicate evaluated to be true, which implies that each honest node j above has received m_j from the client such that $\mathsf{hash}(m_j) = H[j]$. □

Lemma 4. *If an honest node has finished the dispersal phase with H as the output of the RBC, then any honest client can reconstruct the same H after invoking* RETRIEVE*.*

Proof. By Agreement of AVID, all honest nodes eventually finish the dispersal phase once an honest node has finished the dispersal phase. Also, due to Lemma 3 all honest nodes output the same H for RBC when the dispersal phase terminates. Therefore, when an honest client invokes RETRIEVE, all $2t+1$ honest nodes will send the correct $\langle \texttt{HASH}, h'_i, \mathsf{hash}(H) \rangle$ to the client. By OEC (line 17-20) and the collision resistance property of the hash function, the client can successfully decode the same hash vector H. □

Theorem 5 (Availability and Correctness)**.** *If an honest node has finished the dispersal phase, and some honest clients invoke* RETRIEVE*, then they eventually output the same message M'. Furthermore, if an honest client invoked* DISPERSE(M) *and no other client invokes* DISPERSE *on the same instance, then $M' = M$.*

Proof. By Lemma 4, the honest client reconstructs the same hash vector H as the one output by any honest node during the dispersal phase, where H corresponds to some message M'. Moreover, at least $t+1$ honest nodes have received m_j such that $\mathsf{hash}(m_j) = H[j]$. Therefore, when an honest client invokes `RETRIEVE`, all these $t+1$ honest nodes will send the correct $\langle \texttt{SYMBOL}, m_i \rangle$ where $\mathsf{hash}(m_i) = H[i]$ to the retrieving client, enabling it to reconstruct the message M'.

Let $f_u(\cdot)$ denote the polynomial of degree t or less a retrieving client u obtains via interpolation. The client u then uses $f_u(\cdot)$ to recover the message M_u only if $\mathsf{hash}(f_u(i)) = H[i]$ for all $i \in [n]$. Hence, due to collision resistance property of the $\mathsf{hash}(\cdot)$, if two retrieving client, u and v outputs messages $M_u \neq \perp$ and $M_v \neq \perp$, respectively, then $M_u = M_v$.

Also, if any honest retrieving client outputs $\perp$, then every honest retrieving client outputs $\perp$. For the sake of contradiction, let us assume that a retrieving client u outputs $\perp$ but another retrieving client $v \neq u$ outputs $M_v \neq \perp$. Let T_u be the set of indices used by retrieving client u to interpolate f_u. Then, there exists a $k \in [n] \setminus T_u$ such that $H[k] \neq \mathsf{hash}(f_u[k])$. Since retrieving client v outputs $M_v \neq \perp$, this implies $f_u[k] = f_v[k]$ for all $k \in T_u$. However, both f_u and f_v have degree t or less and agrees on $t+1$ distinct points. This implies f_u and f_v matches as polynomial and $f_u[k] = f_v[k]$ for all $k \in [n]$, which is a contradiction.

If an honest dispersing client invoked $\texttt{DISPERSE}(M)$ and no other dispersing client invokes `DISPERSE` on the same instance, by the Termination property of AVID, all honest nodes eventually finish the dispersal phase with RBC output $H = [\mathsf{hash}(m_1), \ldots, \mathsf{hash}(m_n)]$ where $[m_1, \ldots, m_n] = \mathsf{RSEnc}(M, n, t+1)$. Also, from Lemma 4 the retrieving client will receive H during the retrieval phase. Then, by collision-resistant property of the hash function, the honest retrieving client use correct symbol to recover the message, hence will recover and output M. □

Theorem 6 (Performance)**.** *The communication cost of a dispersing client during the dispersal phase is $O(|M|+\kappa n)$ and the total communication cost of the dispersal phase is $O(|M|+\kappa n^2)$. Also, each node incurs a storage cost of $O(|M|/n + \kappa)$. Furthermore, the total communication cost for retrieval at a client is $O(|M|+\kappa n)$.*

Proof. During the dispersal phase, the dispersing client only sends a symbol of size $O(|M|/n)$ to each node and reliably broadcasts a message of size κn. Hence, the total communication cost of the dispersing client is $O(|M|+\kappa n)$ from Lemma 3. Also, each node receives a symbol of size $O(|M|/n)$ and participates in the RBC of a message of size $O(\kappa n)$. Hence, using Lemma 3, the total communication cost of the dispersal phase is $O(|M|+\kappa n^2)$. At the end of the dispersal phase, each node stores two symbol of size $O(|M|/n)$ and $O(\kappa)$, respectively, and a hash output of size κ. Thus, the total storage cost of our AVID protocol is $O(|M|+\kappa n)$.

During retrieval, each node sends at most two symbols of size $O(\kappa)$ and $O(|M|/n)$ to the retrieving client. Hence, the communication cost of a single retrieving client is $O(|M|+\kappa n)$. □

5 Lower Bounds

In this section, we prove communication complexity lower bounds for deterministic protocols that solve RBC and AVID, which have been mentioned in Table 1 and 2. To strengthen the result, the lower bounds for RBC are proven under synchrony. The lower bound proofs are inspired by [14].

5.1 Reliable Broadcast

For any deterministic RBC protocol with input M that tolerates up to $\Theta(n)$ Byzantine nodes, it is straightforward to show a lower bound of $\Omega(n|M|+n^2)$ [26] on the communication cost even under synchrony. The $\Omega(n|M|)$ part is because $O(n)$ honest nodes need to receive the message when the protocol terminates, and the $\Omega(n^2)$ part is due to the classic Dolev-Reischuk lower bound [14]. Therefore, both Das et al. [12] and our BalRBC have near-optimal communication cost.

Next, for any deterministic protocol that solves RBC under synchrony, we will show that $\Omega(|M|+n)$ is a lower bound on the communication cost of any protocol node including the broadcaster, which implies our BalRBC and BalEFRBC has near-optimal per-node cost as well.

Theorem 7. *In any deterministic protocol that solves RBC, for any honest node p, there exists an execution in which p incurs a communication cost of $\Omega(|M|+n)$.*

Proof. We will prove that for any deterministic RBC protocol, all honest nodes incur at least $\Omega(|M|+n)$ communication cost in at least one execution.

The argument for broadcaster is straightforward. First, the broadcaster needs to send at least $\Omega(|M|)$ bits for its input message M. Moreover, the broadcaster has to send messages to at least $t+1$ nodes, otherwise it is possible that no honest node receives any information from the broadcaster, and the Validity property of RBC can be violated. Since $t = \Theta(n)$, we conclude that the broadcaster has to send $\Omega(|M|+n)$ bits.

Consider any non-broadcaster honest node during any failure-free execution where the broadcaster has input M. This honest node needs to output M due to the Validity requirement, so at least $\Omega(|M|)$ bits need to be received.

Let $C_{p,E}$ denote the number of messages an honest node p sends to any node and receives from any honest node during an execution E. We show that $C_{p,E} \geq t/2+1$ for any honest node p in at least one execution E. Otherwise, suppose there exists a RBC protocol where an honest node q has $C_{q,E} \leq t/2$ for any execution E. If q receives no message during the entire execution but other honest nodes output for RBC, due to Totality q eventually outputs as well. Without loss of generality, suppose q outputs 0 in this case. Consider a failure-free execution $E1$ where the honest broadcaster has input 1. By assumption, $C_{q,E1} \leq t/2$. Let S denote the set of nodes that q receives messages from in $E1$. We have $|S| \leq t/2$. Consider execution $E2$ where the honest broadcaster has input 1, and q is Byzantine and remains silent. Since the broadcaster is honest and has input 1, by Validity, all honest nodes output 1 in $E2$. Then, we construct another execution $E3$ same as $E2$ except that the nodes in S are Byzantine and q is now honest. The nodes in S behave identically as in $E2$, except that they send no message to q. By assumption, $C_{q,E3} \leq t/2$. The adversary also corrupts the set of nodes R that q sends messages to in $E3$. This is within the adversary's corruption budget since $|S \cup R| \leq |S|+|R| \leq t$. The Byzantine nodes in R behave identically as in $E2$. Since q receives no message in $E3$, q will output 0 in $E3$ by assumption. Other honest nodes will output 1 in $E3$ since they cannot distinguish $E2, E3$. However, the Agreement property of RBC is then violated. Therefore, we prove that $C_{p,E} \geq t/2+1$ for any honest node p in at least one execution E, which implies the communication cost at any honest node for any RBC protocol is $\Omega(n)$.

Therefore, in any deterministic protocol that solves RBC, for any honest node p, there exists an execution in which p incurs a communication cost of $\Omega(\max\{|M|, n\}) = \Omega(|M|+n)$. □

5.2 Asynchronous Verifiable Information Dispersal

For AVID (or even synchronous VID), the communication cost of the dispersing client during dispersal phase is lower bounded by $\Omega(|M|+n)$ by a similar argument from the above section – the dispersing client needs to send $\Omega(|M|)$ bits and needs to send messages to at least $t+1 = \Omega(n)$ nodes. For the total communication cost during dispersal, we will show the following $\Omega(|M|+n^2)$ lower bound for AVID.

Theorem 8. *Any deterministic protocol that solves AVID must incur a communication cost of $\Omega(|M|+n^2)$ during the dispersal phase in at least one execution.*

Proof. Recall that the communication cost of the dispersing client during dispersal phase is lower bounded by $\Omega(|M|+n)$. Now we will show that any deterministic protocol that solves AVID must uses $\geq (t/2)^2$ messages during the dispersal phase in at least one execution. Suppose it is not true, and there exists a deterministic AVID protocol that uses $< (t/2)^2$ messages during the dispersal phase for all executions. Consider the following two executions.

1. *Execution $E1$:* Let the dispersing client be honest. The adversary corrupts the set of nodes B where $|B| = t/2$. Let A denote the rest of the nodes. For each node $b \in B$, b does not send messages to each other, and ignores the first $t/2$ messages received.

   One observation is that $\exists p \in B$ such that p receives $< t/2$ messages, since the protocol uses $< (t/2)^2$ messages during the dispersal phase and $|B| = t/2$. Also, due to Termination, all honest nodes in A terminate the dispersal phase in $E1$.

2. *Execution $E2$ same as $E1$ except the following differences:* Let $A(P)$ denote the set of nodes that (attempt to) send p messages in $E1$. Since p receives $< t/2$ messages, $|A(p)| < t/2$. The adversary corrupts nodes in $A(p)$ and $B \setminus \{p\}$. The corrupted nodes in $B \setminus \{p\}$ behaves like in $E1$ and ignores all messages from node p. The corrupted nodes in $A(p)$ behave like in $E1$ but do not send messages to p.

Claim: $E1$ and $E2$ are distinguishable to honest nodes in $A \setminus A(p)$, and the honest nodes in $A \setminus A(p)$ will terminate the dispersal phase in $E2$ as well.

The claim can be shown by examining how every node behaves in $E1, E2$. Nodes in $B \setminus \{p\}$ behave the same to all nodes. Nodes in $A(p)$ behave the same to $A \setminus A(p)$. Node p behaves the same since in both executions it does not receive any message from others. Thus the claim is true. □

Now, consider two cases.

- If p never terminates the dispersal phase in $E2$, then Agreement property is violated since other honest nodes terminates the dispersal phase in $E2$, contradiction.
- If p terminates the dispersal phase in $E2$ without receiving any messages, suppose p terminates at time τ. Then consider another execution $E3$ where the dispersing client is Byzantine and remains silent, and all nodes are honest. All messages from any node to p are delayed after τ. Since p cannot distinguish $E2$ and $E3$, it terminates the dispersal phase in $E3$ at time τ as well. Due to the Agreement property, all honest nodes will eventually terminate the dispersal phase in $E3$. Now consider another execution $E4$ where the dispersing client with message M is honest but its messages are delayed, and all nodes are honest. For all honest nodes, they cannot distinguish $E3$ and $E4$ before receiving any message from the dispersing client. Therefore, these honest nodes will terminate the dispersal phase in $E4$ before receiving any message from the dispersing client. In the retrieval phase of $E4$, according to the Availability property, the retrieving client reconstructs some message eventually at some time τ'. Suppose the messages of the dispersing client are delayed beyond time τ' in $E4$. Since no honest node receives any message from the dispersing client, during the retrieval phase of $E4$ the dispersed message cannot be reconstructed, violating the Correctness property of AVID, hence a contradiction.

Hence, any deterministic protocol that solves AVID must uses $\geq (t/2)^2$ messages during the dispersal phase in at least one execution, and thus must incur a communication cost of $\Omega(\max\{|M|+n, (t/2)^2\}) = \Omega(|M|+n^2)$. □

Theorem 9. *Any deterministic protocol that solves AVID must incur a communication cost of $\Omega(|M|+n)$ during the retrieval phase in at least one execution.*

Proof. Consider any execution with an honest retrieving client. The client needs to receive at least $\Omega(|M|)$ bits from the honest nodes to obtain M, and the client needs to send messages to at least $t+1 = \Omega(n)$ nodes for retrieval otherwise it could be the t nodes are Byzantine and ignore the message. Hence, the retrieval phase has cost $\Omega(|M|+n)$. □

6 Related Work

Reliable Broadcast. The problem of reliable broadcast (RBC) was introduced by Bracha [8]. In the same paper, Bracha provided an error-free RBC protocol for a single bit with a communication cost of $O(n^2)$, thus $O(n^2|M|)$ for $|M|$ bits using a naïve approach. Almost two decades later, Cachin and Tessaro [9] improved the cost to $O(n|M|+\kappa n^2 \log n)$ assuming a collision-resistant hash function with κ being the output size of the hash. Hendricks et al. in [19] propose an alternate RBC protocol with a communication cost of $O(n|M|+\kappa n^3)$ using a erasure coding scheme where each element of a codeword can be verified for correctness. Assuming a trusted setup phase, hardness of q-SDH [5, 6] and Decisional Bilinear Diffie-Hellman (DBDH) [7], Nayak et al. [26] reduced the communication cost to $O(n|M|+\kappa n^2)$.

Recently, Das et al. [12] presents a RBC protocol that has a communication cost to $O(n|M|+\kappa n^2)$ assuming only collision-resistant hash function. However, in their RBC protocol, the broadcaster incurs a higher

communication cost than the non-broadcaster nodes. Our protocol maintains the same total communication cost while ensuring that each node, including the broadcaster, incurs asymptotically the same communication cost.

The original RBC protocol due to Bracha [8] is error-free, i.e., it does not require any cryptographic assumptions and is secure against any computationally unbounded adversary in all executions. The error-free RBC protocol due to Patra [27] achieves a total communication cost of $O(n|M|+n^4 \log n)$, and it is later improved to $O(n|M|+n^3 \log n)$ by Nayak et al. [26]. The two protocols above are not balanced; the broadcaster has a cost roughly $O(n)$ higher than other nodes.

Our error-free RBC builds upon the recent result on synchronous error-free Byzantine agreement due to Chen [11]. In particular, our observation is that, with appropriate changes, we can use Chen's protocol to establish the initial condition of the Asynchronous Data Dissemination (ADD) problem introduced by [12]. ADD is a protocol that efficiently disseminates the message from a subset of honest node to all honest nodes in an asynchronous network. Note that, Chen's approach do not rely on any cryptographic assumption and incurs a communication cost of $O(n|M|+n^2 \log n)$. Thus, by combining the modified Chen's protocol along with our balancing technique from §3 and information theoretic Asynchronous Data Dissemination protocol of [12], we get an information-theoretically secure RBC protocol with near-optimal communication cost of $O(n|M|+n^2 \log n)$.

Asynchronous Verifiable Information Dispersal. The first AVID protocol is due to Cachin and Tessaro [9]. In their protocol, during the dispersal phase, each node, including the dispersing client, incurs a communication cost of $O(|M|+\kappa n \log n)$, leading to a total dispersal cost of $O(n|M|+\kappa n^2 \log n)$. This cost arises because every node needs to send a symbol and its associated Merkle path proof to all nodes. Finally, during retrieval, each retrieving client incurs a communication cost of $O(|M|+\kappa n \log n)$. Again, the $\log n$ factor is due to nodes sending Merkle path proofs to the retrieving client.

Hendricks et al. in [19] propose an alternate AVID protocol where during dispersal, the dispersing client incurs a communication cost of $O(|M|+\kappa n^2)$. They improve the communication cost using a non-interactive verification scheme with fingerprinted cross-checksum. In their protocol, only the dispersing client sends the symbols to all nodes, and the nodes perform an RBC on the fingerprinted cross-checksum but not the symbols. As a result, the remaining nodes incur a communication cost of $O(\kappa n^2)$. Hence, the total communication cost of their protocol during the dispersal phase is $O(|M|+\kappa n^3)$. Also, the total retrieval cost for a single retrieving client is $O(|M|+\kappa n^2)$, as each node sends a $O(\kappa n)$ size finger-printed cross-checksum and an encoded symbol to the retrieving client.

Very recently, Yang et al. [30] presents a new AVID protocol in which, during the dispersal phase, the dispersing client incurs a communication cost of $O(|M|+\kappa n \log n)$. Furthermore, the total communication cost of their dispersal phase is $O(|M|+\kappa n^2)$. The main innovation of the AVID protocol of [30] is that they remove the need for nodes to gossip symbols and Merkle path proofs during the dispersal phase. They do so by designing a novel retrieval protocol and a RBC on the root of the associated Merkle tree. Nevertheless, during the dispersal phase, the dispersing client still needs to send a Merkle path proof to every node. Moreover, during retrieval, each node still sends an encoded symbol and the associated Merkle path proof to the retrieving client, leading to a communication cost of $O(|M|+\kappa n \log n)$. Our protocol improves the communication costs of both these steps by a factor of $\log n$ using a vector of hashes instead of a Merkle tree, along with our balanced RBC protocol for long messages.

With trusted setup and assuming hardness of q-SDH [20], the recent work by Alhaddad et al. [3] achieves the dispersing client cost to $O(|M|+\kappa n)$ and the total communication to $O(|M|+\kappa n^2)$ using the KZG [20] polynomial commitment scheme. Our protocol achieves the same cost using only collision-resistant hash function without any trusted setup or additional cryptographic assumptions other than collision-resistant hash functions.

7 Discussion and Conclusion

We have presented two asynchronous Byzantine reliable broadcast (RBC) protocols with a balanced communication cost at all nodes including the broadcaster. The first RBC protocol assumes a collision resistant

hash function and achieves the same total communication complexity as the state-of-the-art, and second RBC protocol is error-free and improves the cost of the state-of-the-art error-free protocol. Our balanced RBC protocol immediately implies a balanced communication cost of asynchronous verifiable/complete secret sharing schemes [12]. We also present an asynchronous verifiable information dispersal (AVID) protocol with improved communication and storage cost. Finally, we present lower bound results on the communication cost of any honest node in any deterministic RBC protocol, and the total communication costs of dispersal and retrieval phases in any deterministic AVID protocol. Our balanced RBC protocol and AVID protocol have near-optimal communication costs. Interesting open problems include designing an AVID protocol that also achieves optimal or near-optimal total storage cost, and designing an information theoretic (or error-free) AVID protocol that has efficient communication cost.

References

[1] Ittai Abraham, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, Gilad Stern, and Alin Tomescu. Reaching consensus for asynchronous distributed key generation. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, page 363–373, 2021.

[2] Ittai Abraham, Kartik Nayak, Ling Ren, and Zhuolun Xiang. Good-case latency of byzantine broadcast: A complete categorization. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing (PODC)*, page 331–341, 2021.

[3] Nicolas Alhaddad, Sisi Duan, Mayank Varia, and Haibin Zhang. Succinct erasure coding proof systems. *Cryptology ePrint Archive*, 2021.

[4] Michael Ben-Or, Ran Canetti, and Oded Goldreich. Asynchronous secure computation. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 52–61, 1993.

[5] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *International conference on the theory and applications of cryptographic techniques*, pages 56–73. Springer, 2004.

[6] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the sdh assumption in bilinear groups. *Journal of cryptology*, 21(2):149–177, 2008.

[7] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *International conference on the theory and application of cryptology and information security*, pages 514–532. Springer, 2001.

[8] Gabriel Bracha. Asynchronous byzantine agreement protocols. *Information and Computation*, 75(2):130–143, 1987.

[9] Christian Cachin and Stefano Tessaro. Asynchronous verifiable information dispersal. In *24th IEEE Symposium on Reliable Distributed Systems (SRDS'05)*, pages 191–201. IEEE, 2005.

[10] Jinyuan Chen. Fundamental limits of byzantine agreement. *arXiv preprint arXiv:2009.10965*, 2020.

[11] Jinyuan Chen. Optimal error-free multi-valued byzantine agreement. In *35th International Symposium on Distributed Computing (DISC 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[12] Sourav Das, Zhuolun Xiang, and Ling Ren. Asynchronous data dissemination and its applications. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 2705–2721, 2021.

[13] Sourav Das, Tom Yurek, Zhuolun Xiang, Andrew Miller, Lefteris Kokoris-Kogias, and Ling Ren. Practical asynchronous distributed key generation. *Cryptology ePrint Archive*, 2021.

[14] Danny Dolev and Rüdiger Reischuk. Bounds on information exchange for byzantine agreement. *Journal of the ACM (JACM)*, 32(1):191–204, 1985.

[15] Sisi Duan, Michael K Reiter, and Haibin Zhang. Beat: Asynchronous bft made practical. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 2028–2041, 2018.

[16] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 1985.

[17] Shuhong Gao. A new algorithm for decoding reed-solomon codes. In *Communications, information and network security*, pages 55–68. Springer, 2003.

[18] Bingyong Guo, Zhenliang Lu, Qiang Tang, Jing Xu, and Zhenfeng Zhang. Dumbo: Faster asynchronous bft protocols. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 803–818, 2020.

[19] James Hendricks, Gregory R Ganger, and Michael K Reiter. Verifying distributed erasure-coded data. In *Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing*, pages 139–146, 2007.

[20] Aniket Kate, Gregory M Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In *International conference on the theory and application of cryptology and information security*, pages 177–194. Springer, 2010.

[21] Eleftherios Kokoris-Kogias, Dahlia Malkhi, and Alexander Spiegelman. Asynchronous distributed key generation for computationally-secure randomness, consensus, and threshold signatures. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1751–1767, 2020.

[22] Donghang Lu, Thomas Yurek, Samarth Kulshreshtha, Rahul Govind, Aniket Kate, and Andrew Miller. Honeybadgermpc and asynchromix: Practical asynchronous mpc and its application to anonymous communication. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 887–903, 2019.

[23] Yuan Lu, Zhenliang Lu, Qiang Tang, and Guiling Wang. Dumbo-mvba: Optimal multi-valued validated asynchronous byzantine agreement, revisited. In *Proceedings of the 39th Symposium on Principles of Distributed Computing*, pages 129–138, 2020.

[24] Florence Jessie MacWilliams and Neil James Alexander Sloane. *The theory of error correcting codes*, volume 16. Elsevier, 1977.

[25] Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. The honey badger of bft protocols. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 31–42, 2016.

[26] Kartik Nayak, Ling Ren, Elaine Shi, Nitin H Vaidya, and Zhuolun Xiang. Improved extension protocols for byzantine broadcast and agreement. In *34th International Symposium on Distributed Computing (DISC 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[27] Arpita Patra. Error-free multi-valued broadcast and byzantine agreement with optimal communication complexity. In *International Conference On Principles Of Distributed Systems*, pages 34–49. Springer, 2011.

[28] Irving S Reed and Gustave Solomon. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 8(2):300–304, 1960.

[29] Lloyd R Welch and Elwyn R Berlekamp. Error correction for algebraic block codes, December 30 1986. US Patent 4,633,470.

[30] Lei Yang, Seo Jin Park, Mohammad Alizadeh, Sreeram Kannan, and David Tse. Dispersedledger: High-throughput byzantine consensus on variable bandwidth networks. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, 2022.

[31] Thomas Yurek, Licheng Luo, Jaiden Fairoze, Aniket Kate, and Andrew Miller. hbacss: How to robustly share many secrets. In *(To appear) Proceedings of the 29th Annual Network and Distributed System Security Symposium*, 2022.

A Balanced Error-Free Reliable Broadcast

In this section, we extend the error-free synchronous Byzantine agreement protocol of [11] to obtain an error-free asynchronous Byzantine reliable broadcast protocol with communication cost $O(n|M|+n^2 \log n)$. Thus, for $|M| \geq O(n \log n)$, our protocol is optimal in terms of communication cost. We also apply the same technique of §3 to balance the cost of broadcaster and other nodes so that the protocol has per-node cost $O(|M|+n \log n)$.

A.1 Design of BalEFRBC

Our error-free RBC has five phases: phase 0 to 4. We summarize our protocol in Algorithm 4 and describe each phase in detail.

Phase 0: The purpose of phase 0 is to let the broadcaster to send its proposal, M, to all nodes. As discussed in §3, if the broadcaster sends its proposal directly to each node, the broadcaster would incur a communication cost of $O(n|M|)$. We adopt the approach we design in §3. More specifically, during phase 0, the broadcaster encodes M using a $(n, t+1)$ Reed-Solomon code. Let $[m_1, m_2, \ldots, m_n]$ be the encoded symbols. The broadcaster then sends the i-th symbol m_i to node i as $\langle \texttt{PROPOSE}, m_i \rangle$ message. Each node i, upon receiving $\langle \texttt{PROPOSE}, m_i \rangle$ message from the broadcaster, sends $\langle \texttt{SHARE}, m_i \rangle$ to all other nodes. Every node then performs information theoretic online error correction using the `SHARE` messages to recover the potential proposal. Let M_i be the proposal node i recovers at the end of phase 0.

Phase 1: During phase 1, each node first encode the proposal it recovered during phase 0 using a (n, k) Reed-Solomon code for $k = \lfloor \frac{t}{5} \rfloor + 1$. Let $[y_1^{(i)}, y_2^{(i)}, \ldots, y_n^{(i)}] := \mathsf{RSEnc}(M_i, n, k)$ be the output of the encoding procedure at node i (line 11). Node i then sends the $\langle \texttt{SYMBOLS}, y_i^{(i)}, y_j^{(i)} \rangle$ to node j for every $j \in [n]$ (line 12). Also, node i upon receiving $\langle \texttt{SYMBOLS}, y_i^{(j)}, y_j^{(j)} \rangle$ from node j adds node j to the set $\mathcal{S}_1^1$ if $(y_i^{(i)}, y_j^{(i)}) = (y_i^{(j)}, y_j^{(j)})$ (line 15-16). Otherwise, node i adds j to the set $\mathcal{S}_0^1$ (line 18). Node i then waits until either $|\mathcal{S}_1^1|$ is greater than or equal to $n-t$, or $|\mathcal{S}_0^1|$ is greater than or equal to $t+1$. The event $|\mathcal{S}_1^1| \geq n-t$ implies that node i received matching symbols from at least $n-t$ nodes. Alternatively, the event $|\mathcal{S}_0^1| \geq t+1$ implies that node i received non-matching symbol from at least $t+1$ nodes.

Upon $|\mathcal{S}_1^1| \geq n-t$, node i sends the message $\langle P1, s_i^1 = 1 \rangle$ to all nodes (line 19-20). Alternatively, if $|\mathcal{S}_0^1| \geq t+1$, node i sends the message $\langle P1, s_i^1 = 0 \rangle$ to all nodes (line 21-22). Finally, for every $\langle P1, s_j^1 = 1 \rangle$ message received from any node j, node i adds node j to the set $\mathcal{S}_1^2$ once it received j's symbols in phase 1 and the symbols match (line 24-27). Otherwise, node i adds the senders of $\langle P1, 0 \rangle$ message to the set $\mathcal{S}_0^2$ (line 29). The reason for waiting to receive node j's symbols in line 25 is that we do not want to add nodes with mismatched symbols in $\mathcal{S}_1^2$, which is crucial for Lemma 5 to hold. Note that it is still possible that an honest node j whose symbols do not match node i, i.e., $j \in \mathcal{S}_0^1$ at node i, sends $s_j^2 = 1$. In our protocol node i ignores all such messages (line 24-27).

Phase 2: During phase 2, if s_i^1 as calculated in phase 1 (line 19-22) is equal to 0, node i sends the $\langle P2, s_i^2 = 0 \rangle$ to every node. Otherwise, if s_i^1 is 1, then depending upon the size of $\mathcal{S}_1^2$ or $\mathcal{S}_0^2$, node i sends the following message. Upon $|\mathcal{S}_1^2| \geq n-t$, node i sends $\langle P2, s_i^2 = 1 \rangle$ to all nodes (line 31-32). Otherwise, upon $|\mathcal{S}_0^2| \geq t+1$,

Algorithm 4 BalEFRBC protocol, code for node i, $i \in [n]$

PHASE 0:

// *only broadcaster node*

input M

Let $[m_1, m_2, \ldots, m_n] := \mathsf{RSEnc}(M, n, t+1)$

send $\langle \texttt{PROPOSE}, m_j \rangle$ to node j for each $j \in [n]$

// *each node* i

Let $M := \perp$

Initialize $\mathcal{S}_0^1, \mathcal{S}_1^1, \mathcal{S}_0^2, \mathcal{S}_1^2, \mathcal{S}_0^3, \mathcal{S}_1^3, \mathcal{S}_0^4, \mathcal{S}_1^4$ to be $\emptyset$

upon receiving the first $\langle \texttt{PROPOSE}, m_i \rangle$ from the broadcaster **do**

  send $\langle \texttt{SHARE}, m_i \rangle$ to all nodes

For the first $\langle \texttt{SHARE}, m_j^* \rangle$ received from node j, add (j, m_j^*) to T // T *initialized as* $\emptyset$

Perform IT-OEC for set T (Algorithm 1)

Let M_i be the returned value of IT-OEC

PHASE 1:

Let $[y_1^{(i)}, y_2^{(i)}, \ldots, y_n^{(i)}] := \mathsf{RSEnc}(M_i, n, k)$

send $\langle \texttt{SYMBOLS}, (y_j^{(i)}, y_i^{(i)}) \rangle$ to node j, $\forall j \in [n]$. // *Exchange symbols*

upon receiving $\langle \texttt{SYMBOLS}, (y_i^{(j)}, y_j^{(j)}) \rangle$ from node j for the first time **do**

  if $(y_i^{(j)}, y_j^{(j)}) = (y_i^{(i)}, y_j^{(i)})$ **then**

    Let $\mathcal{S}_1^1 := \mathcal{S}_1^1 \cup \{j\}$.

  else

    Let $\mathcal{S}_0^1 := \mathcal{S}_0^1 \cup \{j\}$.

upon $|\mathcal{S}_1^1| >= n - t$ **do**

  set $s_i^1 = 1$, send $\langle P1, s_i^1 \rangle$ to all.

upon $|\mathcal{S}_0^1| >= t + 1$ **do**

  set $s_i^1 = 0$, send $\langle P1, s_i^1 \rangle$ to all.

upon receiving $\langle P1, s_j^1 \rangle$ from node j for the first time **do**

  if $s_j^1 = 1$ **then**

    Wait till $j \in \mathcal{S}_0^1 \cup \mathcal{S}_1^1$

    if $j \in \mathcal{S}_1^1$ **then**

      Let $\mathcal{S}_1^2 := \mathcal{S}_1^2 \cup \{j\}$.

  else

    Let $\mathcal{S}_0^2 = \mathcal{S}_0^2 \cup \{j\}$.

PHASE 2:

if $s_i^1 = 1$ **then**

  upon $|\mathcal{S}_1^2| >= n - t$ **do**

    set $s_i^2 = 1$, send $\langle P2, s_i^2 \rangle$ to all.

  upon $|\mathcal{S}_0^2| >= t + 1$ **do**

    set $s_i^2 = 0$, send $\langle P2, s_i^2 \rangle$ to all.

else

  set $s_i^2 = 0$, send $\langle P2, s_i^2 \rangle$ to all

upon receiving $\langle P2, s_j^2 \rangle$ from node j for the first time **do**

  if $s_j^2 = 1$ **then**

    Wait till $j \in \mathcal{S}_0^1 \cup \mathcal{S}_1^1$

    if $j \in \mathcal{S}_1^1$ **then**

      Let $\mathcal{S}_1^3 := \mathcal{S}_1^3 \cup \{j\}$.

  else

    Let $\mathcal{S}_0^3 := \mathcal{S}_0^3 \cup \{j\}$.

PHASE 3:

if $s_i^2 = 1$ **then**

  upon $|\mathcal{S}_1^3| >= n - t$ **do**

    set $s_i^3 = 1$, send $\langle P3, s_i^3 \rangle$ to all.

  upon $|\mathcal{S}_0^3| >= t + 1$ **do**

    set $s_i^3 = 0$, send $\langle P3, s_i^3 \rangle$ to all.

else

  set $s_i^3 = 0$, send $\langle P3, s_i^3 \rangle$ to all.

upon receiving $\langle P3, s_j^3 \rangle$ from node j for the first time **do**

  if $s_j^3 = 1$ **then**

    Let $\mathcal{S}_1^4 := \mathcal{S}_1^4 \cup \{j\}$.

  else

    Let $\mathcal{S}_0^4 = \mathcal{S}_0^4 \cup \{j\}$.

upon $|\mathcal{S}_1^4| >= n - t$ **do**

  send $\langle \texttt{ECHO}, s_i^4 = 1 \rangle$ to all.

upon $|\mathcal{S}_0^4| >= t + 1$ **do**

  send $\langle \texttt{ECHO}, s_i^4 = 0 \rangle$ to all.

upon receiving $2t + 1$ $\langle \texttt{ECHO}, s \rangle$ for matching s and not having sent a `READY` message **do**

  send $\langle \texttt{READY}, s \rangle$ to all

upon receiving $t + 1$ $\langle \texttt{READY}, s \rangle$ for matching s and not having sent a `READY` message **do**

  send $\langle \texttt{READY}, s \rangle$ to all

upon receiving $2t + 1$ $\langle \texttt{READY}, s \rangle$ for matching s **do**

  if $s = 0$ **then**

    output $M = \perp$ and **return**

  start PHASE 4

PHASE 4:

// *only after executing line 67*

Wait till receiving $t + 1$ $\langle \texttt{SYMBOLS}, (y_i^{(j)}, *) \rangle$, $\forall j \in \mathcal{S}_1^4$ // `SYMBOLS` *messages from PHASE 1, and set* $\mathcal{S}_1^4$ *from PHASE 3*

Let $m_i := y_i^{(j)}$

send $\langle \texttt{RECONSTRUCT}, m_i \rangle$ to all

For the first $\langle \texttt{RECONSTRUCT}, m_j^* \rangle$ received from node j, add (j, m_j^*) to T // T *initialized as* $\emptyset$

Perform IT-OEC for set T (Algorithm 1)

Let M_i be the returned value of IT-OEC

output M_i and **return**.

node i sends $\langle P2, s_i^2 = 0\rangle$ to every node (line 33-34). Also, similar to phase 1, for every $\langle P2, s_j^2 = 1\rangle$ message received from any node j, node i adds node j to the set $\mathcal{S}_1^3$ once it received j's symbols in phase 1 and the symbols match (Line 38-41). Otherwise, node i adds the senders of $\langle P2, 0\rangle$ message to the set $\mathcal{S}_0^3$ (line 43).

Phase 3: The first part of phase 3 (line 44-55) is similar to phase 2 (line 30-43), except that any node j that sends $\langle P3, s_i^3 = 1\rangle$ is included in set $\mathcal{S}_1^4$ without the additional checks as in phase 1 and 2.

The remaining steps of phase 3 is analogous to running the 1-bit RBC protocol due to Bracha [8]. Specifically, upon $|\mathcal{S}_1^4| \geq n - t$, node i sends $\langle \texttt{ECHO}, s_i^4 = 1\rangle$ to all nodes (line 56-57). Otherwise, if $|\mathcal{S}_0^4| \geq t + 1$, node i sends $\langle \texttt{ECHO}, s_i^4 = 0\rangle$ to every node (line 58-59). Intuitively, the content of the `ECHO` message (1 or 0) from node i denotes the opinion of node i on whether every node should output $M' \neq \perp$ or $M' = \perp$. Each node upon receiving $2t + 1$ $\langle \texttt{ECHO}, s\rangle$ messages for a matching s, sends the $\langle \texttt{READY}, s\rangle$ message, if it have not sent it already (line 60-61). A node also sends the $\langle \texttt{READY}, s\rangle$ message upon receiving $t+1$ matching $\langle \texttt{READY}, s\rangle$ messages, if have not sent it already (line 62-63). Finally, upon receiving $2t+1$ matching $\langle \texttt{READY}, s\rangle$, if $s = 0$, node i outputs $\perp$ and returns (line 65-66). Otherwise, node i proceeds to phase 4 (line 67).

Phase 4: A node starts phase 4 only after receiving $2t + 1$ $\langle \texttt{READY}, 1\rangle$ messages. During phase 4, each node waits for $t + 1$ matching $\langle \texttt{SYMBOLS}, y_i^{(j)}, *\rangle$ from nodes in $\mathcal{S}_1^4$ (line 69). Recall that $\langle \texttt{SYMBOLS}, y_i^{(j)}, *\rangle$ are sent during phase 1. Let $m_i := y_i^(j)$ be the symbol received in $t + 1$ `SYMBOLS` messages with matching $y_i^(j)$ (line 70). Then, each node sends the message $\langle \texttt{RECONSTRUCT}, m_i\rangle$ to all nodes (line 71). Finally, each node uses the received `RECONSTRUCT` messages to perform OEC and outputs the output of the OEC algorithm (line 72-75).

A.2 Analysis

In this section we will analyze algorithm 4 and show that it implements an error-free RBC protocol for large messages with communication cost of $O(n|M|+n^2 \log n)$ and tolerates up to 1/3-rd Byzantine nodes. Our proof directly uses several Lemmas from [10] and we only provide the lemma statement for those lemmas.

Lemma 5 (Key Lemma)**.** *When any honest node i sends $\langle \texttt{ECHO}, 1\rangle$ at phase 3, all honest nodes in node i's set $\mathcal{S}_1^4$ recovers the same message at the end of phase 0.*

Proof. The proof of this lemma follows directly from the proof of [10, Lemma 3] where we use our proof of Lemma 8 and 9. □

Theorem 10 (Totality and Agreement)**.** *Algorithm 4 guarantees the Totality and Agreement property.*

Proof. Suppose an honest node outputs M', then it has received $2t + 1$ $\langle \texttt{READY}, s\rangle$ messages for matching s. Then since at least $t + 1$ messages above are from honest nodes, every honest node will eventually receive $t + 1$ $\langle \texttt{READY}, s\rangle$ messages. Note that no honest node can send `READY` for any $s' \neq s$, due to the quorum intersection of `ECHO` messages. Thus all honest nodes will send $\langle \texttt{READY}, s\rangle$ and thus receive $2t + 1$ $\langle \texttt{READY}, s\rangle$. If $s = 0$, this implies that every honest will output the default message $M' = \perp$ and return. Otherwise, if $s = 1$, each node will start phase 4. What remains to show is that, during phase 4, each honest node will send a reconstruct message with a correct symbol of the encoding of a unique message M' (line 71) after receiving $t + 1$ matching `SYMBOLS` (line 69), and decode and output M' (line 72-75).

An honest node receiving $2t+1$ $\langle \texttt{READY}, 1\rangle$ implies that at least one honest node sent a $\langle \texttt{ECHO}, 1\rangle$ message. Without loss of generality, let i be the first node that sent an $\langle \texttt{ECHO}, 1\rangle$ message. Observe that node i sends $\langle \texttt{ECHO}, 1\rangle$ only when $|\mathcal{S}_1^4| \geq n - t$ at node i. This means at least $n - 2t \geq t + 1$ nodes in $|\mathcal{S}_1^4|$ are honest and each such node j sent $s_j^3 = 1$ to all nodes. Also, due to Lemma 5, for every pair of honest nodes, they have the same initial message at the end of phase 0, i.e., $j, \ell \in \mathcal{S}_1^4$, $M_j = M_\ell$. Hence, every honest node i will eventually will receive at least $t + 1$ matching $\langle \texttt{SYMBOLS}, (y_i^{(j)}, *)\rangle$ messages from honest nodes in $\mathcal{S}_1^4$. This implies, every honest node will send `RECONSTRUCT` message with correct symbol, and due to properties of OEC, each honest will output the same message M'. □

Theorem 11 (Validity)**.** *Algorithm 4 guarantees the Validity property.*

Proof. When the broadcaster is honest and has input M, due to guarantees of OEC, during phase 0, every honest node will eventually receive M, i.e., $M_i = M_j = M$ for all honest nodes i and j. Since, RSEnc is a deterministic function, for every pair of honest nodes i and j, the tuple of symbols will match, i.e., $(y_i^{(j)}, y_j^{(j)}) = (y_i^{(i)}, y_j^{(i)})$. Since there are at least $n-t$ honest nodes, eventually $|\mathcal{S}_1^1|$ will be greater than or equal to $n-t$ at all honest nodes and every honest node i will send $\langle P1, s_i^2 = 1\rangle$ to others. No honest node will send $\langle P1, s_i^2 = 0\rangle$ since there are at most t Byzantine nodes who may send inconsistent symbols. A similar argument also implies that during phase 2 and 3, each honest node i will send $\langle P2, s_i^3 = 1\rangle$ and $\langle P3, s_i^3 = 1\rangle$, respectively, to all other nodes. Hence, every honest node will eventually send $\langle \texttt{ECHO}, 1\rangle$ and $\langle \texttt{READY}, 1\rangle$ to others and all honest node will start to phase 4. Finally, during phase 4, every honest node sends a valid coded symbol of M in RECONSTRUCT message. Thus, again due to guarantees of OEC, every honest node will reconstruct the message M. □

Theorem 12 (Performance)**.** *For any message M of size $|M|$, the total communication cost of Algorithm 4 is $O(n|M|+n^2\log n)$ bits. Furthermore each node incurs a communication cost of $O(|M|+n\log n)$.*

Proof. During phase 0, the broadcaster sends a symbol to each node, and each node gossips the symbol to every other node. Note that in Reed-Solomon code, each symbol is of size $\max\{|M|/n, \log n\}$ bits. Hence, the communication cost of each node during phase 0 is at most $O(|M|+n\log n)$. Hence, the total communication during phase 0 is at most $O(n|M|+n^2\log n)$. During phase 1, each node sends two symbols (line 12) and a single bit (line 19 and 22) to every other node. Hence, by the same argument as above, the total communication cost of phase 1 is at most $O(n|M|+n^2\log n)$. A node only sends a single bit to other nodes during phase 2. Similarly, during phase 3, each node only sends 1-bit to others and runs a 1-bit Bracha's RBC protocol. Hence, the communication cost of phase 2 and phase 3 is $O(n^2)$. Finally, during phase 4, each node sends a symbol to all other nodes, hence the per node and the total communication cost of phase 4 is $O(|M|+n\log n)$ and $O(n|M|+n^2\log n)$, respectively.

Combining the above, the per node and the total communication cost of Algorithm 4 is $O(|M|+n\log n)$ and $O(n|M|+n^2\log n)$, respectively. □

A.3 Re-proving Lemmas

In this section we will re-prove some lemmas from [10] for our Key Lemma (Lemma 5). Our proofs basically follow the original proof of [10], adapted to our asynchronous RBC protocol.

We will first restate the definitions introduced by Chen [10] in our protocol language.

Notation for groups of nodes. We divide the n-node network into group of nodes. The group definition is based on the values of the messages recovered by nodes at the end of phase 0 and values of the success indicators $\{s_i^j\}_{i=1}^n$ for $j = 1, 2, 3, 4$. Let $\mathcal{F}$ be the group consisting of the indices of all of the dishonest nodes. Note that $|\mathcal{F}| \leq t$. We define the following groups of honest nodes.

$$\mathcal{A}_l \triangleq \{i : M_i = \bar{M}_l,\ i \notin \mathcal{F},\ i \in [n]\}, \quad l \in [\eta] \tag{1}$$

$$\mathcal{A}_l^{[1]} \triangleq \{i : s_i^1 = 1, M_i = \bar{M}_l,\ i \notin \mathcal{F},\ i \in [n]\}, \quad l \in [\eta^{[1]}] \tag{2}$$

$$\mathcal{A}_l^{[2]} \triangleq \{i : s_i^2 = 1, M_i = \bar{M}_l,\ i \notin \mathcal{F},\ i \in [n]\}, \quad l \in [\eta^{[2]}] \tag{3}$$

$$\mathcal{A}_l^{[3]} \triangleq \{i : s_i^3 = 1, M_i = \bar{M}_l,\ i \notin \mathcal{F},\ i \in [n]\}, \quad l \in [\eta^{[3]}] \tag{4}$$

for some different non-empty values $\bar{M}_1, \bar{M}_2, \cdots, \bar{M}_\eta$ and some non-negative integers $\eta, \eta^{[1]}, \eta^{[2]}, \eta^{[3]}$ such that $\eta^{[3]} \leq \eta^{[2]} \leq \eta^{[1]} \leq \eta$. The above definition implies that Group $\mathcal{A}_l$ is a subset of honest nodes who recovered the same message at the end of phase 0. $\mathcal{A}_l^{[1]}$ is a subset of $\mathcal{A}_l$ who have the same non-empty value of updated messages at the end of phase 1. Note that at the end of phase 1, if the updated message of honest node i is non-empty, then it implies that its updated message remains the same as its message after phase 0. Moreover, $s_i^1 = 1$. Similarly, $\mathcal{A}_l^{[2]}$ is a subset of $\mathcal{A}_l^{[1]}$ who have the same non-empty value of updated

messages at the end of Phase 2 for $l \in [\eta^{[2]}]$, while $\mathcal{A}_l^{[3]}$ is a subset of $\mathcal{A}_l^{[2]}$ who have the same non-empty value of updated messages at the end of Phase 3 for $l \in [\eta^{[3]}]$. In our setting, when $1 \leq \eta^{[3]} \leq \eta^{[2]} \leq \eta^{[1]} \leq \eta$, the sets $\mathcal{A}_l, \mathcal{A}_{l_1}^{[1]}, \mathcal{A}_{l_2}^{[2]}, \mathcal{A}_{l_3}^{[3]}$ are all non-empty for any $l \in [\eta], l_1 \in [\eta^{[1]}], l_2 \in [\eta^{[2]}], l_3 \in [\eta^{[3]}]$. Note that $\sum_{l=1}^{\eta} |\mathcal{A}_l| + |\mathcal{F}| = n$.

Let $\mathcal{B}^{[p]}$ defined as

$$\mathcal{B}^{[p]} \triangleq \{i : s_i^p = 0,\ i \notin \mathcal{F},\ i \in [n]\}, \quad p \in \{1, 2, 3\}. \tag{5}$$

Based on our definitions, it holds true that

$$\sum_{l=1}^{\eta^{[p]}} |\mathcal{A}_l^{[p]}| + |\mathcal{B}^{[p]}| + |\mathcal{F}| = n, \quad p \in \{1, 2, 3\}. \tag{6}$$

Throughout our analysis, for any message M, we use $M(\cdot) = \mathsf{RSEnc}(M, n, k)$ to denote the Reed-Solomon encoding of the message M. Moreover, we use $M(i)$ to denote the i-th symbol of $M(\cdot)$. For some $i \in \mathcal{A}_l$, the equality of $\bar{M}_l(i) = \bar{M}_j(i)$ might be satisfied for some j and l. Thus, we further sub-divide the group $\mathcal{A}_l$ the following (possibly overlapping) sub-groups

$$\mathcal{A}_{l,j} \triangleq \{i :\ i \in \mathcal{A}_l, \bar{M}_l(i) = \bar{M}_j(i)\}, \quad j \neq l, j, l \in [\eta] \tag{7}$$

$$\mathcal{A}_{l,l} \triangleq \mathcal{A}_l \setminus \{\cup_{j=1, j\neq l}^{\eta} \mathcal{A}_{l,j}\}, \quad l \in [\eta]. \tag{8}$$

Similarly, Group $\mathcal{A}_l^{[p]}$ can be further divided into some sub-groups defined as

$$\mathcal{A}_{l,j}^{[p]} \triangleq \{i : i \in \mathcal{A}_l^{[p]}, \bar{M}_l(i) = \bar{M}_j(i)\}, \quad j \neq l, j, l \in [\eta^{[p]}] \tag{9}$$

$$\mathcal{A}_{l,l}^{[p]} \triangleq \mathcal{A}_l^{[p]} \setminus \{\cup_{j=1, j\neq l}^{\eta^{[p]}} \mathcal{A}_{l,j}^{[p]}\}, \quad l \in [\eta^{[p]}]. \tag{10}$$

for $p \in \{1, 2, 3\}$.

Notation for graphs. Chen [10] defines a graph $G = (\mathcal{P}, \mathcal{E})$, where $\mathcal{P}$ consists of $n - t$ vertices, i.e., $\mathcal{P} = [n - t]$, and $\mathcal{E}$ is the set of edges. Let $i^* \in \mathcal{P}$, and let $\mathcal{C} \subseteq \mathcal{P} \setminus \{i^*\}$ be a of vertices with $|\mathcal{C}| \geq n - 2t - 1$, such that each vertex in $\mathcal{C}$ is connected with at least $n - 2t$ edges and one of the edges is connected to vertex i^*. We count an edge connecting to itself as an edge as well. For any pair of vertices $i, j \in \mathcal{P}$, we use $E_{i,j} = 1$ (resp. $E_{i,j} = 0$) to indicate that there is an edge (resp. no edge) between vertex i and vertex j. In summary, in G, for a given $i^* \in \mathcal{P} = [n - t]$, the following properties regarding the set C holds.

$$E_{i,i^*} = 1 \quad \forall i \in \mathcal{C} \tag{11}$$

$$\sum_{j \in \mathcal{P}} E_{i,j} \geq n - 2t \quad \forall i \in \mathcal{C} \tag{12}$$

$$|\mathcal{C}| \geq n - 2t - 1 \tag{13}$$

For the graph G, let $\mathcal{D} \subseteq \mathcal{P}$ denote the set of vertices such that each vertex in $\mathcal{D}$ is connected with at least k vertices in $\mathcal{C}$, that is,

$$\mathcal{D} \triangleq \left\{ i :\ \sum_{j \in \mathcal{C}} E_{i,j} \geq k,\ i \in \mathcal{P} \setminus \{i^*\} \right\} \tag{14}$$

where k is the Reed-Solomon encoding parameter. Then, the following lemma provides a result on bounding the size of $\mathcal{D}$.

Lemmas from [10]. Next we will provide Lemmas from [10] that we will directly use later for re-proving Lemmas for our Algorithm 4.

Lemma 6. *For $\mathcal{A}_{l,j}$ and $\mathcal{A}^{[1]}_{l,j}$ defined in (7) and (9), and for $\eta \geq \eta^{[1]} \geq 2$, the following inequalities hold true*

$$|\mathcal{A}_{l,j}|+|\mathcal{A}_{j,l}|<k, \quad \forall j \neq l,\ j,l \in [\eta] \tag{15}$$

$$|\mathcal{A}^{[1]}_{l,j}|+|\mathcal{A}^{[1]}_{j,l}|<k, \quad \forall j \neq l,\ j,l \in [\eta^{[1]}] \tag{16}$$

where k is the Reed-Solomon encoding parameter.

Proof. Refer to [10, Lemma 7] for proof. □

Lemma 7. *For any graph $G = (\mathcal{P}, \mathcal{E})$ specified by (11)-(13) and for the set $\mathcal{D} \subseteq \mathcal{P}$ defined by (14), and given $n \geq 3t + 1$, it holds true that*

$$|\mathcal{D}| \geq n - 9t/4 - 1. \tag{17}$$

Proof. We refer the reader to [10, Lemma 8] for proof. □

Re-proving Lemmas from [10]. We next argue that at the end of phase 2, if there exists 1 or more group of honest nodes with different messages, then it must hold true that the initial size of each group must be at least $n - 9t/4$. More formally,

Lemma 8. *When $\eta^{[2]} \geq 1$, it holds true that $|\mathcal{A}_l| \geq n - 9t/4$, for any $l \in [\eta^{[2]}]$.*

Proof. The proof consists of the following steps, which mostly follows the proof of [10, Lemma 9]:

- Step (a): Transform the network into a graph that is within the family of graphs satisfying (11)-(13) for a fixed i^* in $\mathcal{A}^{[2]}_{l^*}$ and $l^* \in [\eta^{[2]}]$.
- Step (b): Bound the size of a group of honest nodes, denoted by $\mathcal{D}'$ (with the same form as in (14)), using the result of Lemma 7, i.e., $|\mathcal{D}'| \geq n - 9t/4 - 1$.
- Step (c): Argue that every node in $\mathcal{D}'$ has the same initial message as node i^*.
- Step (d): Conclude from Step (c) that $\mathcal{D}'$ is a subset of $\mathcal{A}_{l^*}$, i.e., $\mathcal{D}' \cup \{i^*\} \subseteq \mathcal{A}_{l^*}$ and conclude that the size of $\mathcal{A}_{l^*}$ is bounded by the number determined in Step (b), i.e., $|\mathcal{A}_{l^*}| \geq |\mathcal{D}'|+1 \geq n - 9t/4 - 1 + 1$, for $l^* \in [\eta^{[2]}]$.

Step (a): The first step of the proof is to transform the network into a graph that is within the family of graphs defined above. We will consider the case of $\eta^{[2]} \geq 1$. Recall that, when $\eta^{[2]} \geq 1$, we have $|\mathcal{A}^{[2]}_l| \geq 1$ for any $l \in [\eta^{[2]}]$. Let us consider a fixed i^* for $i^* \in \mathcal{A}^{[2]}_{l^*}$ and $l^* \in [\eta^{[2]}]$. Note that,

$$s^2_{i^*} = 1 \Rightarrow \text{At node } i^*, |\mathcal{S}^2_1| \geq n - t \tag{18}$$

$$\Rightarrow \text{At node } i^*, |\mathcal{S}^2_1 \cap \{\cup_{p=1}^{\eta^{[1]}} \mathcal{A}^{[1]}_p\}| \geq n - 2t \tag{19}$$

$$\Rightarrow \text{At node } i^*, |\mathcal{S}^1_1 \cap \{\cup_{p=1}^{\eta^{[1]}} \mathcal{A}^{[1]}_p\}| \geq n - 2t \tag{20}$$

the last implication follows from the fact that $j \in \mathcal{S}^2_1 \Rightarrow j \in \mathcal{S}^1_1$, $\forall j \in \cup_{p=1}^{\eta^{[1]}} \mathcal{A}^{[1]}_p$, which holds due to the check in line 25-26 in Algorithm 4.

Let $\mathcal{C}'$ be defined as follows;

$$\mathcal{C}' \triangleq \{\mathcal{S}^1_1 \text{ at node } i^* \cap \{\cup_{p=1}^{\eta^{[1]}} \mathcal{A}^{[1]}_p\}\} \setminus \{i^*\} \Rightarrow |\mathcal{C}'| \geq n - 2t - 1 \tag{21}$$

Moreover, since $\mathcal{C}'$ is a subset of $\cup_{p=1}^{\eta^{[1]}}\mathcal{A}_p^{[1]}$, it implies

$$s_j^1 = 1, \forall j \in \mathcal{C}' \Rightarrow \text{At every node} j \in \mathcal{C}', |\mathcal{S}_1^1| \geq n - t \tag{22}$$

$$\Rightarrow \text{At every node} j \in \mathcal{C}', |\mathcal{S}_1^1 \cap \{\cup_{l=1}^{\eta}\mathcal{A}_l\}| \geq n - 2t \tag{23}$$

$$\tag{24}$$

In other words, for any $j \in \mathcal{C}'$, node j receives at least $n - 2t$ number of matched observations from honest nodes during phase 1. Let us define a subset of $\{\cup_{l=1}^{\eta}\mathcal{A}_l\} \setminus \{i^*\}$ of honest nodes as

$$\mathcal{D}' \triangleq \{p : p \text{ received matching } \texttt{SYMBOLS} \text{ message from at least } k \text{ nodes in } \mathcal{C}' \text{ and } p \in \{\cup_{l=1}^{\eta}\mathcal{A}_l\} \setminus \{i^*\}\} \tag{25}$$

where k is the Reed-Solomon encoding parameter.

Now we map the network into a graph by considering the honest nodes as the vertices and considering the link indicators as edges. Let $\mathcal{P} \triangleq \cup_{l=1}^{\eta}\mathcal{A}_l$. Let $\mathcal{E}$ consists of $E_{i,j}$ for all $i, j \in \mathcal{P}$ such that $E_{i,j} = 1$ if $i \in \mathcal{S}_1^1$ at node j. Note that $i \in \mathcal{S}_1^1$ at node j implies eventually $j \in \mathcal{S}_1^1$ at node i. Let $G = (\mathcal{P}, \mathcal{E})$ be a graph, then $\mathcal{C}' \subseteq \mathcal{P}' \setminus \{i^*\}$ be as defined equation (21). It is easy to see that the graph G falls into a family of graphs satisfying (11)-(13).

The step (b), (c), and (d) of our proof is identical to the proof of steps (b), (c), and (d) of [10, Lemma 9]. Thus we omit them for brevity. □

Next we provide a lemma that will be used later for the analysis of the proposed protocol.

Lemma 9. *In algorithm 4 with $n \geq 3t + 1$, if $\eta^{[1]} = 2$ then it holds true that $\eta^{[3]} \leq 1$.*

Proof. The proof mostly follows the proof of [10, Lemma 10]. Given $\eta^{[1]} = 2$, the definitions in (1)-(10) imply that

$$\mathcal{A}_1^{[1]} = \{i : s_i^1 = 1, M_i = \bar{M}_1,\ i \notin \mathcal{F},\ i \in [n]\} \tag{26}$$

$$\mathcal{A}_2^{[1]} = \{i : s_i^1 = 1, M_i = \bar{M}_2,\ i \notin \mathcal{F},\ i \in [n]\} \tag{27}$$

$$\mathcal{A}_{1,2}^{[1]} = \{i :\ i \in \mathcal{A}_1^{[1]},\ \bar{M}_1(i) = \bar{M}_2(i)\} \tag{28}$$

$$\mathcal{A}_{1,1}^{[1]} = \mathcal{A}_1^{[1]} \setminus \mathcal{A}_{1,2}^{[1]} = \{i :\ i \in \mathcal{A}_1^{[1]}, \bar{M}_1(i) \neq \bar{M}_2(i)\} \tag{29}$$

$$\mathcal{A}_{2,1}^{[1]} = \{i :\ i \in \mathcal{A}_2^{[1]}, \bar{M}_2(i) = \bar{M}_1(i)\} \tag{30}$$

$$\mathcal{A}_{2,2}^{[1]} = \mathcal{A}_2^{[1]} \setminus \mathcal{A}_{2,1}^{[1]} = \{i :\ i \in \mathcal{A}_2^{[1]},\ \bar{M}_2(i) \neq \bar{M}_1(i)\} \tag{31}$$

$$\mathcal{B}^{[1]} = \{i : s_i^1 = 0,\ i \notin \mathcal{F},\ i \in [n]\} = \{i : i \in [n],\ i \notin \mathcal{F} \cup \mathcal{A}_1^{[1]} \cup \mathcal{A}_2^{[1]}\}. \tag{32}$$

In the following we will complete the proof by focusing on the following three cases

$$\text{Case 1:} \quad |\mathcal{A}_1^{[1]}| + |\mathcal{B}^{[1]}| \geq t + 1 \tag{33}$$

$$|\mathcal{A}_2^{[1]}| + |\mathcal{B}^{[1]}| < t + 1 \tag{34}$$

$$\text{Case 2:} \quad |\mathcal{A}_1^{[1]}| + |\mathcal{B}^{[1]}| < t + 1 \tag{35}$$

$$|\mathcal{A}_2^{[1]}| + |\mathcal{B}^{[1]}| \geq t + 1 \tag{36}$$

$$\text{Case 3:} \quad |\mathcal{A}_1^{[1]}| + |\mathcal{B}^{[1]}| \geq t + 1 \tag{37}$$

$$|\mathcal{A}_2^{[1]}| + |\mathcal{B}^{[1]}| \geq t + 1. \tag{38}$$

Note that the following case

$$\text{Case 4:} \quad |\mathcal{A}_1^{[1]}|+|\mathcal{B}^{[1]}|<t+1 \tag{39}$$

$$|\mathcal{A}_2^{[1]}|+|\mathcal{B}^{[1]}|<t+1 \tag{40}$$

does not exist. See [10] for its proof.

Case 1: Recall that in the first step of Phase 2, due to the check of line 25-26 in Algorithm 4, each node $i \in \mathcal{A}_{2,2}$ with $s_i^1 = 1$ eventually adds a node j to $\mathcal{S}_0^2$ for every $j \in \mathcal{S}_0^1$. Moreover, by definition eventually every node in $\mathcal{B}^{[1]} \cup \mathcal{A}_1$ will be added to $\mathcal{S}_0^1$ at each node $i \in \mathcal{A}_{2,2}$ because $s_i^1 = 0, \quad \forall i \in \mathcal{B}^{[1]}$ and nodes $\mathcal{A}_1$ and $\mathcal{A}_{2,2}$ have mismatched symbols. Since we assume in (33), that $|\mathcal{A}_1^{[1]}|+|\mathcal{B}^{[1]}| \geq t+1$, it implies that eventually during Phase 2, every node $i \in \mathcal{A}_{2,2}$ sets

$$s_i^2 = 0, \quad \forall i \in \mathcal{A}_{2,2}^{[1]} \tag{41}$$

With the outcome in (41) and after exchanging the success indicators, eventually, the set of $\mathcal{A}_{2,2}^{[1]}$, as well as $\mathcal{B}^{[1]}$, will be in the list of $\mathcal{S}_0$ at each honest node. Note that a subset of $\mathcal{A}_{2,1}^{[1]}$, i.e., $\mathcal{A}_{2,1}^{[1]} \cap \{p : s_p^2 = 1\}$) may still be in list of $\mathcal{S}_1^2$. Below we will argue that the complete set of $\mathcal{A}_{2,1}^{[1]}$ will be in the list of $\mathcal{S}_0^3$.

During Phase 3, a node $i \in \mathcal{A}_{2,1}$ and with $s_i^3 = 1$, eventually adds node j to $\mathcal{S}_0^3$ for each $j \in \mathcal{A}_{2,2}^{[1]} \cup \mathcal{B}^{[1]}$. This holds due to the check in line 39-40 of Algorithm 4. It is also true that

$$\bar{M}_i(j) \neq \bar{M}_j(j), \quad \forall j \in \mathcal{A}_{1,1}^{[1]}, \ i \in \mathcal{A}_{2,1}^{[1]} \tag{42}$$

Note that, the size of $\mathcal{A}_{1,1}^{[1]} \cup \mathcal{A}_{2,2}^{[1]} \cup \mathcal{B}^{[1]}$ can be bounded by

$$|\mathcal{A}_{1,1}^{[1]} \cup \mathcal{A}_{2,2}^{[1]} \cup \mathcal{B}^{[1]}| = |\mathcal{A}_{1,1}^{[1]}|+|\mathcal{A}_{2,2}^{[1]}|+|\mathcal{B}^{[1]}| \tag{43}$$

$$= n - |\mathcal{F}| - |\mathcal{A}_{1,2}^{[1]}| - |\mathcal{A}_{2,1}^{[1]}| \tag{44}$$

$$\geq n - |\mathcal{F}| - (k-1) \tag{45}$$

$$\geq 2t + 1 - (k-1) \tag{46}$$

$$\geq t+1 \tag{47}$$

where (43) uses the disjoint property between $\mathcal{A}_{1,1}^{[1]}$, $\mathcal{A}_{2,2}^{[1]}$ and $\mathcal{B}^{[1]}$; (44) is from (6) and the disjoint property between $\mathcal{A}_{1,1}^{[1]}$, $\mathcal{A}_{1,2}^{[1]}$, $\mathcal{A}_{2,1}^{[1]}$, $\mathcal{A}_{2,2}^{[1]}$ and $\mathcal{B}^{[1]}$; (45) follows from Lemma 6, which implies that $|\mathcal{A}_{1,2}^{[1]}|+|\mathcal{A}_{2,1}^{[1]}| < k$ (or equivalently $|\mathcal{A}_{1,2}^{[1]}|+|\mathcal{A}_{2,1}^{[1]}| \leq k-1$); (46) uses the condition that $n \geq 3t+1$ and $|\mathcal{F}| = t$; (47) results from the fact that $t \geq k-1$ based on our design of k. Hence, for every node $i \in \mathcal{A}_{2,1}$, during phase 3, eventually $|\mathcal{S}_0^3| \geq t+1$. Therefore, during phase 3, the node i updates its success indicator $s_i^3 = 0$. Since, $\mathcal{A}_{2,1}^{[1]} \cap \{p : s_p^2 = 0\} \subseteq \mathcal{S}_0$, this implies that eventually $\mathcal{A}_2^{[1]} \subseteq \mathcal{S}_0^3$. Stating differently, at the end of Phase 3 of there exists *at most* 1 group of honest nodes, where the honest nodes within this group have the same non-empty updated message (with success indicators as ones), and the honest nodes outside this group have the same empty updated message (with success indicators as zeros), that is, $\eta^{[3]} \leq 1$, for Case 1.

Case 2: Due to the symmetry between Case 1 and Case 2, one can follow from the proof steps for Case 1 and interchange the roles of Groups $\mathcal{A}_1$ and $\mathcal{A}_2$ (as well as the roles of Groups $\mathcal{A}_1^{[p]}$ and $\mathcal{A}_2^{[p]}$ accordingly for $p \in \{1,2,3\}$), to show for Case 2 that at the end of Phase 3 it is that $\mathcal{A}_1^{[1]} \subseteq \mathcal{S}_0$.

Case 3: Follows from the analysis of case 1 and the argument presented in [10]. □